

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-195755

(43)Date of publication of application : 09.07.2003

(51)Int.Cl.

G09C 1/00

(21)Application number : 2002-299549

(71)Applicant : MATSUSHITA ELECTRIC IND CO LTD

(22)Date of filing : 11.10.2002

(72)Inventor : FUDA YUICHI  
OMORI MOTOJI  
YOKOTA KAORU  
TATEBAYASHI MAKOTO

(30)Priority

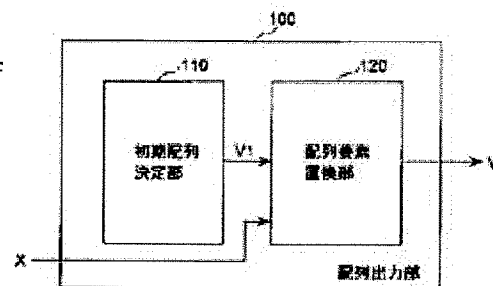
Priority number : 2001321651 Priority date : 19.10.2001 Priority country : JP

### (54) ARRAY OUTPUT DEVICE, ARRAY OUTPUT METHOD, ENCRYPTION DEVICE AND DECRYPTION DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide an array output device that outputs a well-balanced n- dimensional array based on an integer value such as an output value of a hash function value without using so much memory.

SOLUTION: The array output device is provided with an initial array decision unit 110 that tentatively decides an initial decision array V1 having n1 piece of 1, n2 piece of -1 and (n-n1-2) piece of 0 as its array element, and an array element replacement unit 120 that changes the array element of the initial decision array V1 decided by the initial array decision unit 110 based on an input integer X and that outputs the array V.





(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2003-195755

(P2003-195755A)

(43) 公開日 平成15年7月9日 (2003.7.9)

(51) Int.Cl.<sup>7</sup>

G 0 9 C 1/00

識別記号

6 2 0

F I

G 0 9 C 1/00

テマコード\* (参考)

6 2 0 Z 5 J 1 0 4

審査請求 未請求 請求項の数32 O L (全 26 頁)

(21) 出願番号 特願2002-299549 (P2002-299549)

(22) 出願日 平成14年10月11日 (2002.10.11)

(31) 優先権主張番号 特願2001-321651 (P2001-321651)

(32) 優先日 平成13年10月19日 (2001.10.19)

(33) 優先権主張国 日本 (J P)

(71) 出願人 000005821

松下電器産業株式会社

大阪府門真市大字門真1006番地

(72) 発明者 布田 裕一

大阪府門真市大字門真1006番地 松下電器  
産業株式会社内

(72) 発明者 大森 基司

大阪府門真市大字門真1006番地 松下電器  
産業株式会社内

(74) 代理人 100109210

弁理士 新居 広守

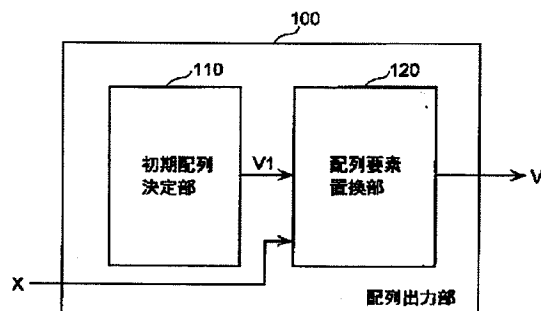
最終頁に続く

(54) 【発明の名称】 配列出力装置、配列出力方法、暗号化装置、および復号化装置

(57) 【要約】

【課題】 多くのメモリを使用することなく、ハッシュ関数値の出力値などの整数値に基づいて一様に  $n$  次元の配列を出力する配列出力装置を提供する。

【解決手段】  $n_1$  個の配列要素が1であり、 $n_2$  個の配列要素が-1であり、 $(n - n_1 - n_2)$  個の配列要素が0である初期決定配列  $V_1$  を暫定的に決定する初期配列決定部110と、入力された整数  $X$  に基づいて、初期配列決定部110が決定した初期決定配列  $V_1$  の配列要素を改編し、配列  $V$  を出力する配列要素置換部120とを備える。





## 【特許請求の範囲】

【請求項1】 入力された整数に依存して、K値の整数の組み合わせからなるn次元の様々な配列を出力する配列出力装置であって、

初期配列を暫定的に決定する初期配列決定手段と、  
前記入力された整数に基づいて、前記初期配列決定手段が決定した前記初期配列の配列要素を改編する改編手段とを備えることを特徴とする配列出力装置。

【請求項2】 前記初期配列決定手段は、K値の整数の組み合わせからなるn次元の配列の1つを初期配列として暫定的に決定し、

前記改編手段は、前記入力された整数に基づいて、前記初期配列決定手段が決定した前記初期配列の配列要素を置換し出力することを特徴とする請求項1記載の配列出力装置。

【請求項3】 前記初期配列決定手段は、K値のうちの同じ整数がそれぞれ連続した位置に配置されている配列を初期配列として決定することを特徴とする請求項2記載の配列出力装置。

【請求項4】 前記n次元の配列は、n1個の配列要素が整数P1、n2個の配列要素が整数P2、・・・、nk個の配列要素が整数PkであるK値の整数の組み合わせからなることを特徴とする請求項3記載の配列出力装置。

【請求項5】 前記改編手段は、  
前記入力された整数を所定の整数で除算し剰余を求める除算部と、

前記除算部が求めた剰余に基づいて、前記初期配列の配列要素を置換する置換部とを備えることを特徴とする請求項4に記載の配列出力装置。

【請求項6】 前記除算部は、前記入力された整数を所定の整数で除算したときの商を被除数としてさらに所定の整数で除算することを繰り返し、

前記置換部は、前記除算部が除算毎に求める剰余に基づいて、前記初期配列の配列要素を順次置換することを特徴とする請求項5記載の配列出力装置。

【請求項7】 前記置換部は、前記初期配列の配列要素のうち、前記除算部が除算したときの除数J番目の配列要素と、前記除算部が除算したときの剰余R+1番目の配列要素とを置換し、さらに置換された配列においても前記除算部が繰り返す除算毎に求める除数および剰余に基づいて同様に順次置換することを特徴とする請求項6記載の配列出力装置。

【請求項8】 前記除算部は、除数をnから2までの各場合において順に除算を行い、

前記置換部は、前記除算部が行う除算毎に前記初期配列を順次置換することを特徴とする請求項7記載の配列出力装置。

【請求項9】 前記n次元の配列は、1、-1、0の3値の組み合わせからなる配列であることを特徴とする請

求項8記載の配列出力装置。

【請求項10】 前記初期配列決定手段は、すべての配列要素がK値のうちのいずれかの整数P3である配列を初期配列として暫定的に決定し、

前記改編手段は、前記初期配列決定手段が決定した前記初期配列における整数P3の配列要素のうち前記入力された整数に基づく位置の配列要素をK値の他の整数P1に置き換えて出力することを特徴とする請求項1記載の配列出力装置。

【請求項11】 前記n次元の配列は、n1個の配列要素が整数P1、n2個の配列要素が整数P2、・・・、nk個の配列要素が整数PkであるK値の整数の組み合わせからなることを特徴とする請求項10記載の配列出力装置。

【請求項12】 前記改編手段は、

前記入力された整数を所定の整数で除算し剰余を求める除算部と、

前記初期配列における整数P3の配列要素のうち前記除算部が求めた剰余に基づく位置の配列要素を整数P1に置き換える整数配置部とを備えることを特徴とする請求項11に記載の配列出力装置。

【請求項13】 前記除算部は、前記入力された整数を所定の整数で除算したときの商を被除数としてさらに所定の整数で除算することを繰り返し、

前記整数配置部は、整数P1がn1個となるまで前記初期配列における整数P3の配列要素のうち前記除算部が除算毎に求める剰余に基づく位置の配列要素を整数P1に順次置き換えることを特徴とする請求項12記載の配列出力装置。

【請求項14】 前記除算部は、除数をnから減少させた各場合において順に除算を行い、

前記整数配置部は、前記初期配列における整数P3の配列要素のうち、前記除算部が除算したときの剰余R+1番目の配列要素を整数P1に順次置き換えることを特徴とする請求項13記載の配列出力装置。

【請求項15】 前記改編手段はさらに、前記整数配置部が前記初期配列のn1個の配列要素を整数P1に置き換えた配列における整数P3の配列要素のうち、さらにn2個の配列要素を前記除算部が求める剰余に基づいてK値のうちの整数P2に置き換える第2の整数配置部を備えることを特徴とする請求項13に記載の配列出力装置。

【請求項16】 前記n次元の配列は3値の組み合わせからなる配列であり、前記整数P1、整数P2、整数P3は、それぞれ1、-1、0のうちのいずれかであることを特徴とする請求項15記載の配列出力装置。

【請求項17】 前記入力された整数は複数のビット情報で示されており、

前記改編手段は、前記入力された整数を、所定数のビット情報からなる個々の分割情報に分割する分割部と、



前記初期配列における整数P3の配列要素のうち、前記分割情報に基づく位置の配列要素をK値の他の整数P1に置き換える第3の整数配置部とを備えることを特徴とする請求項10記載の配列出力装置。

【請求項18】 前記n次元の配列は、n1個の配列要素が整数P1、n2個の配列要素が整数P2、・・・、nk個の配列要素が整数PkであるK値の整数の組み合わせからなることを特徴とする請求項17記載の配列出力装置。

【請求項19】 前記第3の整数配置部は、前記初期配列における整数P3の配列要素のうち、前記個々の分割情報が示す整数に基づく位置の配列要素をK値の他の整数P1に順次置き換えることを特徴とする請求項18記載の配列出力装置。

【請求項20】 前記第3の整数配置部は、前記初期配列における整数P3の配列要素のうちn1個の配列要素が整数P1となるまで、個々の分割情報が示す整数を順に加算した累積値に基づく位置の配列要素を順次整数P1に置き換えることを特徴とする請求項19記載の配列出力装置。

【請求項21】 前記改編手段はさらに、前記第3の整数配置部が前記初期配列のn1個の配列要素を整数P1に置き換えた配列における整数P3の配列要素のうち、さらにn2個の配列要素を前記分割情報に基づいてK値のうちの整数P2に置き換える第4の整数配置部を備えることを特徴とする請求項20に記載の配列出力装置。

【請求項22】 前記n次元の配列は3値の組み合わせからなる配列であり、前記整数P1、整数P2、整数P3はそれぞれ、1、-1、0のうちのいずれかであることを特徴とする請求項21記載の配列出力装置。

【請求項23】 入力された整数に依存して、K値の整数の組み合わせからなるn次元の様々な配列を出力する配列出力方法であって、

初期配列を暫定的に決定する初期配列決定ステップと、前記入力された整数に基づいて、前記初期配列決定ステップで決定した前記初期配列の配列要素を改編する改編ステップとを備えることを特徴とする配列出力方法。

【請求項24】 前記初期配列決定ステップは、K値の整数の組み合わせからなるn次元の配列の1つを初期配列として暫定的に決定し、

前記改編ステップは、前記入力された整数に基づいて、前記初期配列決定ステップで決定した前記初期配列の配列要素を置換し出力することを特徴とする請求項23記載の配列出力方法。

【請求項25】 前記改編ステップは、前記入力された整数を所定の整数で除算し剰余を求める除算ステップと、前記除算ステップで求めた剰余に基づいて、前記初期配列の配列要素を置換する置換ステップとを備えることを特徴とする請求項24に記載の配列出力方法。

【請求項26】 前記初期配列決定ステップは、すべての配列要素がK値のうちのいずれかの整数P3である配列を初期配列として暫定的に決定し、

前記改編ステップは、前記初期配列決定ステップで決定した前記初期配列における整数P3の配列要素のうち前記入力された整数に基づく位置の配列要素をK値の他の整数P1に置き換えて出力することを特徴とする請求項23記載の配列出力方法。

【請求項27】 前記改編ステップは、前記入力された整数を所定の整数で除算し剰余を求める除算ステップと、

前記初期配列における整数P3の配列要素のうち前記除算ステップで求めた剰余に基づく位置の配列要素を整数P1に置き換える整数配置ステップとを備えることを特徴とする請求項26に記載の配列出力方法。

【請求項28】 入力された整数に依存して、K値の整数の組み合わせからなるn次元の様々な配列を出力する配列出力方法のためのプログラムであって、

初期配列を暫定的に決定する初期配列決定ステップと、前記入力された整数に基づいて、前記初期配列決定ステップで決定した前記初期配列の配列要素を改編する改編ステップとをコンピュータに実行させることを特徴とするプログラム。

【請求項29】 前記初期配列決定ステップは、K値の整数の組み合わせからなるn次元の配列の1つを初期配列として暫定的に決定し、

前記改編ステップは、前記入力された整数に基づいて、前記初期配列決定ステップで決定した前記初期配列の配列要素を置換し出力することを特徴とする請求項28記載のプログラム。

【請求項30】 前記初期配列決定ステップは、すべての配列要素がK値のうちのいずれかの整数P3である配列を初期配列として暫定的に決定し、

前記改編ステップは、前記初期配列決定ステップで決定した前記初期配列における整数P3の配列要素のうち前記入力された整数に基づく位置の配列要素をK値の他の整数P1に置き換えて出力することを特徴とする請求項28記載のプログラム。

【請求項31】 メッセージを暗号化する暗号化装置であって、

メッセージを一方関数で演算し、その結果を関数値として出力する関数値出力手段と、

初期配列を暫定的に決定する初期配列決定手段、および前記関数値出力手段が出力する関数値に基づいて、前記初期配列決定手段が決定した前記初期配列の要素を改編する改編手段を備え、前記関数値に依存して、K値の整数の組み合わせからなるn次元の様々な配列を出力する配列出力手段と、

前記配列出力手段が出力する配列を基に、暗号文を生成する暗号文生成手段とを備えることを特徴とする暗号化



装置。

【請求項32】 暗号文を復号化して元のメッセージを出力する復号化装置であって、

暗号文を復号し、元のメッセージに対応する復号値を出力する復号手段と、

前記復号手段が出力する復号値を一方関数で演算し、その結果を関数値として出力する第2の関数値出力手段と、

初期配列を暫定的に決定する初期配列決定手段、および前記第2の関数値出力手段が出力する関数値に基づい

て、前記初期配列決定手段が決定した前記初期配列の要素を改編する改編手段を備え、前記関数値に依存して、K値の整数の組み合わせからなるn次元の様々な配列を出力する配列出力手段と、

前記配列出力手段が出力する配列を基に、チェック用暗号文を生成する暗号文生成手段と、

前記暗号文と前記チェック用暗号文が一致するか否かを判定し、一致するときに前記復号手段が出力する前記復号値に対して所定の処理を施し元のメッセージを出力する出力手段とを備えることを特徴とする復号化装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、整数値を配列に変換する配列出力装置に関するものであり、特に、情報セキュリティ技術としての暗号技術、誤り訂正技術、およびデジタル署名技術に利用される配列出力装置に関するものである。

【0002】

【従来の技術】秘密通信方式とは、特定の通信相手以外に通信内容を漏らすことなく通信を行う方式である。またデジタル署名方式とは、通信相手に通信内容の正当性を示したり、通信元がその本人であることを証明したりする通信方式である。この署名方式には公開鍵暗号と呼ばれる暗号方式を用いる。公開鍵暗号は通信相手が多数の時、通信相手ごとに異なる暗号鍵を容易に管理するための方式であり、多数の通信相手と通信を行うのに不可欠な基盤技術である。簡単に説明すると、これは暗号化鍵と復号化鍵が異なり、復号化鍵は秘密にするが、暗号化鍵を公開する方式である。公開鍵暗号については、岡本 龍明、山本 博資、“現代暗号”、産業図書、1997（非特許文献1と呼ぶ）が詳しい。

【0003】この公開鍵暗号の1つとして、NTRU暗号と呼ばれる暗号がある。このNTRU暗号は、楕円曲線暗号に比べ暗号化のコードサイズが小さく、家電用機器で使用される非力なCPUでも実装可能であり、将来有望な暗号方式である。

【0004】このNTRU暗号については、Jeffrey Hoffstein, Jill Pipher and Joseph H. Silverman, “NTRU: A ring based public

key cryptosystem”, Lecture Notes in Computer Science, 1423, pp. 267-288, Springer-Verlag, 1998（非特許文献2と呼ぶ）に詳しく述べられている。

【0005】ここで、NTRU暗号について説明する。一般的にすべての多項式 $f(X)$ は、 $f(X) = f_0 + f_1 X + f_2 X^2 + \dots + f_{N-1} X^{N-1} \pmod{(X^N - 1)}$ で表される。

【0006】以下、多項式 $f(X)$ をn次元のベクトルであるベクトル $(f_0, f_1, f_2, \dots, f_{N-1})$ と対応つけて表す。また、このn次元のベクトルのうち、n1個が1、n2個が-1、その他の $(n - n1 - n2)$ 個が0であるベクトルを $L(n, n1, n2)$ と表す。

【0007】このNTRU暗号では、秘密にされる復号化鍵（以下、秘密鍵と称する） $f(v)$ 、 $Fp(v)$ は、次の式で表される。尚、以下 $f(v)$ 、 $Fp(v)$ などの $(v)$ を付した符号は、多項式を表している。秘密鍵 $f(v) \in$ 多項式の集合 $L_f$ 、多項式の集合 $L_f = L(263, 51, 50)$

秘密鍵 $Fp(v) =$ 秘密鍵 $f(v)^{-1} \pmod{p}$ すなわち、多項式の集合 $L_f$ は、係数 $f_0, f_1, f_2, \dots, f_{N-1}$ のうち1が51個、-1が50個、0が162個である多項式の集合であり、秘密鍵 $f$ はこの多項式の集合 $L_f$ に属する多項式である。また $p$ は整数であり、例えば3である。

【0008】一方、公開される暗号化鍵（以下、公開鍵と称する） $h(v)$ は、次の式で表される。

公開鍵 $h(v) =$ 秘密鍵 $f(v)^{-1} \times$ 多項式 $g(v) \pmod{q}$

多項式 $g(v) \in$ 多項式の集合 $L_g = L(263, 24, 24)$

ここで、公開鍵 $h(v)$ は多項式である。また、 $q$ は例えば $2^7$ の整数である。

【0009】NTRU暗号では、この公開鍵 $h(v)$ を用いて次式を基に暗号化される。この暗号化において、メッセージ $m1(v)$ の入力に対して、暗号文 $e1(v)$ が出力される。

暗号文 $e1(v) = p \times$ 多項式 $\phi(v) \times$ 公開鍵 $h(v) + m1(v) \pmod{q}$

多項式 $\phi(v) \in$ 多項式の集合 $L_\phi = L(263, 16, 16)$

ここで、多項式 $\phi(v)$ は多項式の集合 $L_\phi$ からランダムに選ばれる。

【0010】一方、暗号文 $e1(v)$ は、上記秘密鍵 $f(v)$ 、 $Fp(v)$ を用いて次の2段階で復号化され、メッセージ $m1'(v)$ が得られる。

(1)  $a(v) =$ 秘密鍵 $f(v) \times$ 暗号文 $e1(v) \pmod{q}$

(2)  $m1'(v) =$ 秘密鍵 $Fp(v) \times a(v) \pmod{m}$



od p)

【0011】ところで、暗号解読の方法には、受動的攻撃と能動的攻撃があり、RSA暗号、ElGamal暗号、NTRU暗号などの暗号では、受動的攻撃のみを考慮して作られている。受動的攻撃、能動的攻撃、RSA暗号、ElGamal暗号については、「非特許文献1」に詳しく述べられている。

【0012】また最近では、一般の暗号方式を、どのような攻撃に対しても安全性を高くする暗号アルゴリズム改良技術である安全性証明スキームに変換する安全性証明スキーム化技術が提案されている。

【0013】その安全性証明スキーム化技術に、FOSRTと呼ばれるハッシュ関数を用いる手法がある。

【0014】FOSRT及び、NTRU暗号にFOSRTを適用する点については、Jeffrey Hoffman and Joseph H. Silverman, "Protecting NTRU Against Chosen Ciphertext and Reaction Attacks", NTRUCryptosystems Technical Report #016, 2000 (非特許文献3と呼ぶ)、ハッシュ関数については、「非特許文献1」に詳しく述べられている。

【0015】このFOSRTの具体的な方法を次に説明する。このFOSRTによる暗号化は、次の3段階で行われ、メッセージMの入力に対して、暗号文Eが出力される。

(第1段階) メッセージMに乱数R1が連結され、連結されたメッセージM||R1を得る。

(第2段階) ハッシュ関数を基に、メッセージM||R1のハッシュ関数値haを得る。  
 $ha = H(M||R1)$

(第3段階) メッセージM||R1とハッシュ関数値haを暗号化のアルゴリズムを基に暗号化し、暗号文Eを得る。

$E = Enc(M||R1, ha)$

【0016】次に、FOSRTによる復号化を説明する。

(第1段階) 暗号文Eを復号化して、メッセージM||R1'を得る。

$Dec(E) = M||R1'$

(第2段階) ハッシュ関数を基に、メッセージM||R1'のハッシュ関数値ha'を得る。 $ha' = H(M||R1')$

(第3段階) メッセージM||R1'、ハッシュ関数値ha'、および上記暗号化と同一のアルゴリズムを基に暗号化し、暗号文E'を得る。

$E' = Enc(M||R1', ha')$

(第4段階) 暗号文Eと暗号文E'が一致しない場合は出力なし、一致する場合は、メッセージM||R1'をメ

ッセージM'と乱数R1'に分解し、所望のメッセージM'を得る。このFOSRTをNTRU暗号に適用したときの、暗号化と復号化の方法を次に示す。

【0017】暗号化は3段階について行われ、入力されたメッセージM(v)に対して暗号文e(v)を出力する。(第1段階) メッセージM(v)にランダムなベクトルR(v)を連結して、メッセージm(v)を得る。 $m(v) = M(v) || R(v)$  (第2段階) ハッシュ関数を基に、メッセージm(v)のハッシュ関数値H(m(v))を計算する。(第3段階) 暗号文e(v) =  $p \cdot H(m(v)) \times \text{公開鍵} h(v) + m(v) \pmod{q}$ の式を基に暗号文e(v)を得る。

【0018】一方、暗号文e(v)の復号化は次の5段階で行われる。(第1段階)  $a(v) = \text{秘密鍵} f(v) \times \text{暗号文} e(v) \pmod{q}$ を基に多項式a(v)を求める。(第2段階)  $m'(v) = \text{秘密鍵} Fp(v) \times a(v) \pmod{p}$ を基にメッセージm'(v)を求める。(第3段階) メッセージm'(v)のハッシュ関数値H(m'(v))を計算し、暗号文e'(v) =  $p \cdot H(m'(v)) \times \text{公開鍵} h(v) + m'(v) \pmod{q}$ を基に、暗号文e'(v)を求める。(第4段階) 暗号文e'(v)と暗号文e(v)が一致するかをチェックする。(第5段階) 暗号文e'(v)と暗号文e(v)が一致する場合に、 $m'(v) = M'(v) || R'(v)$  (M'(v)は復号されたメッセージ、R'(v)はランダムなベクトル)に分解し、メッセージM'(v)を出力する。

【0019】このようにFOSRTをNTRU暗号に適用したときの暗号化、および復号化において、ハッシュ関数値H(m(v))、H(m'(v))は、例えばL(263, 16, 16)で表される多項式の集合Lφに属する必要がある。

【0020】この多項式の集合Lφは、その係数f<sub>0</sub>, f<sub>1</sub>, f<sub>2</sub>, ..., f<sub>N-1</sub>のうち1が16個、-1が16個、0が231個であるベクトルの集合に対応づけられる。そのため、ハッシュ関数値に対応させて、この16個の1、16個の-1、231個の0の3値からなるn次元配列を求める必要がある。

【0021】しかし、FOSRTをNTRU暗号に適用したときの暗号化、および復号化において、このハッシュ関数値H(m(v))、H(m'(v))は整数値となる。そのため、NTRU暗号にFOSRTを適用するために、ハッシュ関数値H(m(v))、H(m'(v))に基づく、n1個の要素が1、n2個の要素が-1、その他の要素が0であるn次元配列を求めなければならない(ここで、n、n1、n2は正の整数である)。

【0022】ここで、ハッシュ関数値H(m(v))、H(m'(v))に基づくn次元の配列を求める方法において、以下に示すような条件を満たす必要がある。



(1) 同じ入力に対して、いつも同じ出力が対応する。  
 (2) 入力と出力の分布が偏らない。(1)は、同じ入力に対して、異なる値を出力しないことである。(2)は、入力に対して、ある出力値だけ、頻繁に現れることのないということである。NTRU暗号にFOSRTを適用する場合は、出力の $n$ 次元配列を送信者、受信者共に、作成可能でないと復号が不可能であるため、(1)を満たさなければ、暗号方式自体が成り立たない。また、(2)を満たさなければ、ハッシュ関数値の出力値に基づいて一様に配列が出力されないため、ハッシュ関数の入力に対する出力の分布の一様性が保持されない。従って、ハッシュ関数の安全性を落とすことになり、FOSRTを適用した時のNTRU暗号の安全性が低下することになる。

【0023】ここで、1個の要素が1、 $n-2$ 個の要素が-1、その他の要素が0である $n$ 次元の配列を求める自明な方法について述べる。図19は、 $n$ 次元の配列を求める方法を示すフローチャートである。この変換方法は、入力をハッシュ関数値としての整数 $X$ と、 $n-1$ 、および $n-2$ とし、 $n-1$ 個の要素が1、 $n-2$ 個の要素が-1、その他の $(n-n-1-n-2)$ 個の要素が0である $n$ 次元の配列 $VJ$ を出力する。以下では、配列 $VJ$ の(左から) $i$ 番目の要素を $VJ[i]$ (但し $i$ は1から $n$ までの整数)とする。

【0024】まず、配列 $VJ$ を全ての要素が0の配列とする(ステップS901)。次に、カウンタ $c1$ のカウンタ値 $c1$ を1とする(ステップS902)。次に、 $VJ[c1]=1$ とする(ステップS903)。次に、カウンタ $c1$ のカウンタ値 $c1$ をカウントアップする(ステップS904)。次に、カウンタ値 $c1 > n-1$ であるかを判定し(ステップS905)、カウンタ値 $c1 > n-1$ でない場合(ステップS905のNo)、再度、 $VJ[c1]=1$ の処理を行う(ステップS903)。

【0025】一方、カウンタ値 $c1 > n-1$ の場合(ステップS905のYes)、 $VJ[c1]=-1$ とする(ステップS906)。次に、カウンタ値 $c1$ をカウントアップする( $c1 \leftarrow c1+1$ )(ステップS907)。次に、カウンタ値 $c1 > n-1+n-2$ であるかを判定し(ステップS908)、カウンタ値 $c1 > n-1+n-2$ でない場合(ステップS908のNo)、再度、 $VJ[c1]=-1$ の処理を行う(ステップS906)。

【0026】一方、カウンタ値 $c1 > n-1+n-2$ の場合(ステップS908のYes)、配列 $VJ$ を出力し、処理を終了する。

【0027】この方法では、出力される配列 $VJ$ は、入力される整数 $X$ に依らず、はじめの $n-1$ 個の要素が1、その後の $n-2$ 個の要素が-1、その他の要素が0の配列になっている。

【0028】一方、秘密通信方式として、送信したいメッセージを鍵で暗号化し、同じ鍵を用いて復号する共通

鍵暗号方式がある。共通鍵暗号方式の中には、データの置換操作で暗号文を作成する方法がある。例えば、以下のような方法である。

【0029】この置換方法は、入力を配列 $m[1]$ 、 $m[2]$ 、 $\dots$ 、 $m[n]$ と、鍵 $Ke$ (正の整数)とし、暗号文 $e[1]$ 、 $e[2]$ 、 $\dots$ 、 $e[n]$ を出力する。以下では、予め2次元テーブル $Tab[j][i]$ ( $1 \leq j \leq n!$ 、 $1 \leq i \leq n$ )を持っているものとする。

【0030】まず、カウンタ $c'$ のカウンタ値 $c$ を1に設定する。次に、 $e[Tab[K][c]]$ に $m[c]$ を代入する。この処理は、カウンタ $c'$ のカウンタ値 $c$ が $n$ となるまで行う。そして、カウンタ $c'$ のカウンタ値 $c$ が $n$ となると、暗号文 $e$ を出力して、処理を終了する。このような置換方法を上述したハッシュ関数値に基づいた $n$ 次元の配列を求める方法に適用することも考えられるが、 $n \times n!$ 個のテーブルが必要となる。

【0031】

【非特許文献1】岡本 龍明、山本 博資、“現代暗号”、産業図書、1997

【0032】

【非特許文献2】Jeffrey Hoffstein, Jill Pipher and Joseph H. Silverman, “NTRU: A ring based public key cryptosystem”, Lecture Notes in Computer Science, 1423, pp. 267-288, Springer-Verlag, 1998

【0033】

【非特許文献3】Jeffrey Hoffstein and Joseph H. Silverman, “Protecting NTRU Against Chosen Ciphertext and Reaction Attacks”, NTRU Cryptosystems Technical Report #016, 2000

【0034】

【発明が解決しようとする課題】しかしながら、上述した $n$ 次元の配列を求める自明な方法では、出力が1種類に偏っているため、上述した条件(2)(入力と出力の分布が偏らない)を満たさない。この時、FOSRTを適用したことの効果なくなる上に、受動攻撃に対しても安全性が弱くなってしまふ。従って、この方法を用いて、NTRU暗号にFOSRTを適用した時のNTRU暗号の安全性が低下するという問題がある。

【0035】また、上述した共通鍵暗号として使用される置換方法を、 $n$ 次元配列を求める時に適用しても、メモリーテーブルを使用するため、多くのメモリーを必要とするという問題がある。

【0036】本発明は、上述した問題点を鑑みて行われ



たもので、多くのメモリを使用することなく、ハッシュ関数値の出力値などの整数値に基づいて一様に $n$ 次元の配列を出力する配列出力装置を提供することを目的とする。

【0037】

【課題を解決するための手段】上記目的を達成するために、本発明に係る配列出力装置は、入力された整数に依存して、 $K$ 値の整数の組み合わせからなる $n$ 次元の様々な配列を出力する配列出力装置であって、初期配列を暫定的に決定する初期配列決定手段と、前記入力された整数

に基づいて、前記初期配列決定手段が決定した前記初期配列の配列要素を改編する改編手段とを備えることを特徴とする。

【0038】また、前記初期配列決定手段は、 $K$ 値の整数の組み合わせからなる $n$ 次元の配列の1つを初期配列として暫定的に決定し、前記改編手段は、前記入力された整数に基づいて、前記初期配列決定手段が決定した前記初期配列の配列要素を置換し出力する構成としてもよい。

【0039】また、前記初期配列決定手段は、すべての配列要素が $K$ 値のうちのいずれかの整数 $P$ 3である配列を初期配列として暫定的に決定し、前記改編手段は、前記初期配列決定手段が決定した前記初期配列における整数 $P$ 3の配列要素のうち前記入力された整数に基づく位置の配列要素を $K$ 値の他の整数 $P$ 1に置き換えて出力する構成としてもよい。

【0040】また、前記入力された整数は複数のビット情報で示されており、前記改編手段は、前記入力された整数を、所定数のビット情報からなる個々の分割情報に分割する分割部と、前記初期配列における整数 $P$ 3の配列要素のうち、前記分割情報に基づく位置の配列要素を $K$ 値の他の整数 $P$ 1に置き換える第3の整数配置部とを備える構成としてもよい。

【0041】また、入力された整数に依存して、 $K$ 値の整数の組み合わせからなる $n$ 次元の様々な配列を出力する配列出力方法であって、初期配列を暫定的に決定する初期配列決定ステップと、前記入力された整数に基づいて、前記初期配列決定ステップで決定した前記初期配列の配列要素を改編する改編ステップとを備えることを特徴とする。

【0042】また、メッセージを暗号化する暗号化装置であって、メッセージを一方関数で演算し、その結果を関数値として出力する関数値出力手段と、初期配列を暫定的に決定する初期配列決定手段、および前記関数値出力手段が出力する関数値に基づいて、前記初期配列決定手段が決定した前記初期配列の要素を改編する改編手段を備え、前記関数値に依存して、 $K$ 値の整数の組み合わせからなる $n$ 次元の様々な配列を出力する配列出力手段と、前記配列出力手段が出力する配列を基に、暗号文を生成する暗号文生成手段とを備えることを特徴とす

る。

【0043】

【発明の実施の形態】（実施の形態1）以下、本発明における第1の実施の形態に係る暗号化装置について図面を用いて説明する。図1(a)は本発明における第1の実施の形態に係る暗号化装置の構成を示すブロック図である。

【0044】暗号化装置10は、乱数生成部20と、連結部30と、ハッシュ関数部40と、配列出力部100と、暗号文生成部50とを備えており、取得したメッセージ $M(v)$ 、および公開鍵 $h(v)$ を基に暗号文 $e(v)$ を生成し出力する。以下、メッセージ $M(v)$ 、公開鍵 $h(v)$ 、暗号文 $e(v)$ などの $(v)$ を付した符号は多項式を示している。また、従来例と同じものには同じ符号を付している。

【0045】この暗号化装置10を構成する乱数生成部20と、連結部30と、ハッシュ関数部40と、配列出力部100と、暗号文生成部50は、それぞれマイクロコンピュータのソフトウェアにより処理を実行しており、CPUやメモリなどを用いて処理が行われる。

【0046】この暗号化装置10は、従来例において説明したように、NTRU暗号に安全性証明スキームのFOSRTを適用して、NTRU暗号の安全性をさらに高めた暗号を生成する。乱数生成部20は、ランダムなベクトル $R(v)$ を生成する。

【0047】連結部30は、乱数生成部20が生成したベクトル $R(v)$ とメッセージ $M(v)$ とを連結して、メッセージ $m(v)$ を生成し、ハッシュ関数部40と暗号文生成部50に出力する。

【0048】ハッシュ関数部40は、メッセージ $m(v)$ を一方関数であるハッシュ関数で演算しハッシュ関数値 $H(m)$ を求め、配列出力部100に出力する。ここで、ハッシュ関数は一方関数であり、ハッシュ関数部40より出力されるハッシュ関数値 $H(m)$ は整数であり、以下、整数 $X$ という。

【0049】配列出力部100は、ハッシュ関数部40より出力される整数 $X$ に基づいた $n$ 次元の配列 $V$ を生成し暗号文生成部50に出力する。

【0050】暗号文生成部50は、配列出力部100から出力される $n$ 次元の配列 $V$ に対応する多項式 $\phi(v)$ と、連結部30からのメッセージ $m(v)$ と、公開鍵 $h(v)$ と、暗号文 $e(v) = p$ 多項式 $\phi(v) \times$ 公開鍵 $h(v) + m(v) \pmod{q}$ の式とを基に暗号文 $e(v)$ を生成し、外部に出力する。 $p$ 、 $q$ は整数であり、 $p$ は例えば3、 $q$ は例えば $2^7$ である。

【0051】一方、図1(b)は本発明における第1の実施の形態に係る復号化装置の構成を示すブロック図である。

【0052】この復号化装置15は、復号部25と、ハッシュ関数部45と、配列出力部105と、暗号文生成



部55と、判定部65と、分割部35とを備えており、入力された暗号文 $e(v)$ 、秘密鍵 $f(v)$ 、 $Fp(v)$ 、および公開鍵 $h(v)$ を基に、暗号化装置10が暗号化した暗号文 $e(v)$ を復号し、元のメッセージ $M'(v)$ を出力する。

【0053】この復号化装置15を構成する復号部25と、ハッシュ関数部45と、配列出力部105と、暗号文生成部55と、判定部65と、分割部35は、それぞれマイクロコンピュータのソフトウェアにより処理を実行しており、CPUやメモリなどを用いて処理を行う。

【0054】復号部25は、次式を基に入力された暗号文 $e(v)$ より元のメッセージに対応する復号値であるメッセージ $m'(v)$ を得る。

$$a(v) = \text{秘密鍵 } f(v) \times \text{暗号文 } e(v) \pmod{q}$$

$$m'(v) = \text{秘密鍵 } Fp(v) \times a(v) \pmod{p}$$

【0055】ハッシュ関数部45は、メッセージ $m'(v)$ のハッシュ関数値 $H(m')$ を演算し、配列出力部105に出力する。ハッシュ関数部45より出力されるハッシュ関数値 $H(m')$ は整数であり、以下、整数 $X'$ という。

【0056】配列出力部105は、ハッシュ関数部45より出力される整数 $X'$ に基づいて $n$ 次元の配列 $V'$ を生成し暗号文生成部55に出力する。

【0057】暗号文生成部55は、配列出力部105から出力される $n$ 次元の配列 $V'$ に対応する多項式 $\phi'(v)$ と、復号部25からのメッセージ $m'(v)$ と、公開鍵 $h(v)$ と、暗号文 $e'(v) = p$ 多項式 $\phi'(v) \times \text{公開鍵 } h(v) + m'(v) \pmod{q}$ の式とを基に暗号文 $e'(v)$ を生成し、判定部65に出力する。

【0058】判定部65は、暗号文 $e(v)$ と、暗号文 $e'(v)$ とを入力し、両者が一致するかを判定し、一致したと判定するとメッセージ $m'(v)$ を分割部35に出力する。

【0059】分割部35は、判定部65からのメッセージ $m'(v)$ をメッセージ $M'(v)$ とランダムなベクトル $R'(v)$ に分解し、元のメッセージ $M'(v)$ を出力する。

【0060】次に、暗号化装置10における配列出力部100について図面を用いて説明する。尚、復号化装置15における配列出力部105は、配列出力部100と同様の構成であり、同様の動作をするため、説明を省略する。

【0061】図2は、配列出力部100の構成を示すブロック図である。この配列出力部100は、整数 $X$ を入力とし、 $L(n, n1, n2)$ に属する配列 $V$ を出力する。ここで、 $L(n, n1, n2)$ は、 $n1$ 個の要素が1、 $n2$ 個の要素が $-1$ 、その他の $(n - n1 - n2)$

個の要素が0である $n$ 次元配列全体であり、この $n$ 、 $n1$ 、 $n2$ は配列出力部100に予め設定されているものとする。配列出力部100は、初期配列決定部110と配列要素置換部120からなる。

【0062】初期配列決定部110は配列 $V$ の初期決定を行うものであり、以下のような初期決定配列 $V1$ を出力する。

$$(1) V1[i] = 1 \quad (1 \leq i \leq n1)$$

$$(2) V1[i] = -1 \quad (n1 + 1 \leq i \leq n1 + n2)$$

$$(3) V1[i] = 0 \quad (n1 + n2 + 1 \leq i \leq n)$$

【0063】ここで、 $V1[i]$ を初期決定配列 $V1$ の(左から) $i$ 番目( $i$ は1から $n$ までの整数)の配列要素(要素)とする。

【0064】図3は、配列出力部100による各段階での配列 $V$ の配列状態を示す図である。図3では、例えば $L(8, 3, 2)$ の場合( $n=8$ ,  $n1=3$ ,  $n2=2$ )における配列 $V$ の各配列状態を示している。図3において、1番上の配列状態は、初期配列決定部110により決定された初期決定配列 $V1$ を示している。

【0065】配列要素置換部120は、初期配列決定部110より出力された初期決定配列 $V1$ と、整数 $X$ を入力とし、 $n1$ 個の要素が1、 $n2$ 個の要素が $-1$ 、その他の $(n - n1 - n2)$ 個の要素が0である $n$ 次元の配列 $V$ を出力する。すなわち、初期配列決定部110が決定した初期決定配列 $V1$ の配列要素を改編する。

【0066】以下、配列要素置換部120の行う処理を説明する。図4は、配列要素置換部120の行う処理を示すフローチャートである。以下では、カウンタ $c'$ の値をカウンタ値 $c$ とする。

【0067】まず、配列要素置換部120は、整数 $X$ を変数 $Y$ に、初期決定配列 $V1$ を配列 $V$ に代入する。即ち、全ての $i$ ( $i$ は1から $n$ までの整数)に対して、 $V1[i]$ を $V[i]$ に代入する(ステップS101)。

【0068】次に、配列要素置換部120は、カウンタ $c'$ のカウント値 $c$ を $n$ に設定する(ステップS102)。

【0069】次に、配列要素置換部120は、変数 $Y$ をカウンタ値 $c$ で割った商を $S$ 、余り(剰余)を $R$ とする(ステップS103)。

【0070】次に、配列要素置換部120は、 $tmp \leftarrow V[c]$ とする(ステップS104)。すなわち、レジスタ $tmp$ に配列 $V$ の $c$ 番目の要素 $V[c]$ を代入する。

【0071】次に、配列要素置換部120は、 $V[c] \leftarrow V[R+1]$ とする(ステップS105)。すなわち、配列 $V$ の $(R+1)$ 番目の要素を配列 $V$ の $c$ 番目の要素に代入する。

【0072】次に、配列要素置換部120は、 $V[R+1] \leftarrow tmp$ とする(ステップS106)。すなわち、レジスタ $tmp$ の内容を配列 $V$ の $(R+1)$ 番目の要素



に代入する。

【0073】この $tmp = V[c]$ 、 $V[c] = V[R+1]$ 、 $V[R+1] = tmp$ の処理(ステップS104からステップS106)により、配列Vのc番目の要素と(R+1)番目の要素が入れ替わる。

【0074】次に、配列要素置換部120は、変数Yに商Sを代入する(ステップS107)。次に、配列要素置換部120は、カウント値cが2であるか否かを判定し(ステップS108)、カウント値cが2であると判定した場合(ステップS108のYes)、配列要素置換部120は、配列Vを出力して(ステップS110)、処理を終了する。

【0075】一方、配列要素置換部120は、カウント値cが2でないと判定した場合(ステップS108のNo)、カウント値cをカウントダウン( $c \leftarrow c-1$ )して(ステップS109)、再度、変数Yをカウント値cで割った商をS、余りをRとする処理(ステップS103)に移る。

【0076】このように、配列要素置換部120は、カウンタc'のカウント値cがnから2の各場合において、商S、余りRを求める処理(ステップS103)からカウント値cをカウントダウンする処理(ステップS109)までの各ステップの処理を繰り返す。これにより、配列Vの各要素が入れ替わり、カウント値c=2での処理が終了すると、配列要素置換部120は配列Vを出力する。

【0077】ここで、配列出力部100全体の動作を説明する。配列出力部100は、まず初期配列決定部110が、はじめのn1個の要素が1、続くn2個の要素が-1、その他の要素が0であるn次元配列の初期決定配列V1を決定して、配列要素置換部120に出力する。

【0078】次に配列要素置換部120は、初期配列決定部110から出力された初期決定配列V1と、配列出力部100に入力された整数Xを取得し、初期決定配列V1の各要素を、整数Xを基に入れ替えて、n1個の要素が1、n2個の要素が-1、その他の要素が0であるn次元の配列Vを出力する。

【0079】この配列出力部100の動作を、具体例を用いて説明する。この配列出力部100の動作を説明する際、例えば実際に入力される整数 $X = 39356$ 、 $n = 8$ 、 $n1 = 3$ 、 $n2 = 2$ を用いて説明する。

【0080】まず、初期配列決定部110は、 $n = 8$ 、 $n1 = 3$ 、 $n2 = 2$ を満たす初期決定配列V1を決定し、配列要素置換部120に出力する(図3参照)。そして、配列要素置換部120は、入力された整数Xを変数Yに代入し、変数 $Y = 39356$ とし、配列Vに初期決定配列V1を代入する(ステップS101)。

【0081】次に、配列要素置換部120は、カウント値c=8とする(ステップS102)。次に、配列要素置換部120は、商Sと余りRを演算し、商 $S = 491$

9、余り $R = 4$ とする(ステップS103)。

【0082】次に、配列要素置換部120は、配列Vの(R+1)番目、すなわち5番目の要素である「-1」と8番目の要素である「0」を入れ替え(ステップS104～ステップS106)、図3の配列V2に示す配列状態とする。次に、配列要素置換部120は、変数Yに4919を代入する(ステップS107)。

【0083】そして、カウント値c=8でありカウント値c=2でないため(ステップS108のNo)、配列要素置換部120は、カウント値cをカウントダウンしてカウント値c=7とする(ステップS109)。

【0084】そして、配列要素置換部120は、再度、変数 $Y = 4919$ 、カウント値c=7より、商Sおよび余りRを演算し、商 $S = 702$ 、余り $R = 5$ とする(ステップS103)。

【0085】次に、配列要素置換部120は、配列Vの左から6番目の要素である「0」と7番目の要素である「0」を入れ替え(ステップS104～ステップS106)、図3の配列V3に示す配列状態とする。

【0086】このように、配列要素置換部120は、カウント値cがnから2であるそれぞれの場合において、変数Yをカウント値cで割ったときの商Sと余りRに基づいて、配列Vの要素の入れ替えを行い(ステップS103→ステップS109の処理の繰り返し)、初期決定配列V1の各要素の入れ替え処理を行う。そして、カウント値c=2である場合の上記処理が終了すると、配列要素置換部120は配列Vを出力する。

【0087】ここで、配列出力部100から出力される配列Vが入力される整数Xに基づいて一様に分布していることを説明する。配列出力部100は、 $0 \leq X \leq (n! - 1)$ (但し $n!$ は「nの階乗」を示す。具体的には、 $n! = n \times (n-1) \times \dots \times 2 \times 1$ である。)を満たす整数Xに基づいて一様に配列L(n, n1, n2)を出力している。以下では、 $X$ を $0 \leq X \leq (n! - 1)$ を満たすXに限定して、説明を行う。

【0088】配列要素置換部120では初期決定配列V1の要素の入れ替えを行っている。以下では、配列要素置換部120が行う初期決定配列V1の要素の入れ替えについて具体的にみていく。また以降の説明を分かり易くするために、上述の配列要素置換部120の構成において、カウント値cがi( $i = n, n-1, n-2, \dots, 3, 2$ )の時、 $R \leftarrow Y \bmod c$ (ステップS103)の処理による余りを $R_i$ とする。

【0089】全体の説明の流れは、以下の通りである。  
(1) 入れ替え処理が重複しないことについて説明する。

(2) (1)の結果を利用して、整数Xと配列要素置換部120における処理内容が1対1に対応していることを説明する。

(3) 同じ配列を出力する整数Xが何種類あるかを計算



する。

この(3)の計算において配列Vに依らず、整数Xの種類数が同じであることを示すことが可能であるため、入力に対して出力が一樣に分布することがいえる。

【0090】まず(1)の「入れ替え処理が重複しないことについて」について説明する。以下では、まず、同じ値に対する入れ替えが複数回起こらない、即ち、入れ替え処理の終わった値は、その処理以降の入れ替えの対象にならないことについて説明する。

【0091】・カウンタc'のカウンタ値cがnの時  
ステップS104、ステップS105、ステップS106の動作(以下、「カウンタ値cがnの場合の入れ替え処理」と呼ぶ)が意味するところは、初期決定配列V1のn番目の要素V1[n]と、(R<sub>n+1</sub>)番目の要素V1[R<sub>n+1</sub>]を入れ替えていることである。ここで、R<sub>n</sub>は、変数Yをnで割った時の余りであるので、0からn-1までの値を取りうる。

【0092】・カウンタ値cがn-1の時  
この時は、カウンタ値cがnである時の入れ替え処理を終えた状態における配列Vのn-1番目の要素と、(R<sub>n-1</sub>+1)番目の要素を入れ替えている。ここで、R<sub>n-1</sub>は、変数Yをカウンタの値(n-1)で割った時の余りであるので、0からn-2までの値を取りうる。ゆえに、カウンタ値cがn-1である時の入れ替え処理を終えた状態の配列Vの(n-1)番目の要素は、カウンタ値cがnである時の入れ替え処理を終えた時点の配列Vの(R<sub>n-1</sub>+1)番目の要素になっている。また、カウンタ値cがnである時の入れ替え処理を終えた状態の配列Vのn番目の要素は、入れ替え対象となっていないことに注意する。

【0093】・カウンタ値cがn-2以下の時  
カウンタ値cがn-2以下の場合について考える。この場合、カウンタ値cがn-1の時と同様である。カウンタ値cがiの時の入れ替え処理を終えた状態の配列Vのi番目の要素は、カウンタ値cが(i+1)の時の入れ替え処理を終えた状態の配列Vの(R<sub>i+1</sub>)番目の要素になっている。ここで、カウンタ値cが(i+1)の時の入れ替え処理は、カウンタ値cがiの時の入れ替え処理の前に行われることを注意しておく。またこれ以降、配列Vのi番目の要素は、他の値と入れ替えられることはない。なぜならばj ≤ i-1とすると、R<sub>j</sub>は変数Yをjで割った時の余りであるから、R<sub>j</sub> ≤ j-1を満たす。よって、(R<sub>j</sub>+1) ≤ i-1 < iとなるからである。

【0094】配列要素置換部120の処理が終了した後、すなわち、カウンタ値cがnから2までの全ての処理を終えた状態での配列Vの各要素は、初期決定配列V1の各要素を入れ替えたものになっている。

【0095】次に、配列要素置換部120の処理の一意性について説明する。以下では、上記の結果(入れ替え

が重複しない)を利用して、整数Xと配列要素置換部120における処理内容が1対1に対応していることを説明していく。

【0096】・整数Xと数列の対応について

整数Xを0 ≤ X ≤ (n!-1)を満たす整数とする。この時、「カウンタc'」のカウンタ値cがcの時のステップS103における変数Yの値をY<sub>c</sub>、商Sの値をS<sub>c</sub>、余りRの値をR<sub>c</sub>とする(R<sub>c</sub>の値については先に定義したものと同一である)。この時、ステップS103で行われていることを式で表現すると、Y<sub>c</sub> = c × S<sub>c</sub> + R<sub>c</sub>となる。但し、0 ≤ R<sub>c</sub> ≤ (c-1)である。またステップS107で行われていることを式で表現すると、Y<sub>(c-1)}</sub> = S<sub>c</sub>となる。

【0097】これを、c=n、(n-1)、…、2の場合に当てはめて整理すると、X=Y<sub>n</sub>であることに注意して、

$$X = (n \times (n-1) \times \dots \times 3) \times R_{n-2} + (n \times (n-1) \times \dots \times 4) \times R_{n-3} + \dots + (n \times (n-1)) \times R_{n-2} + n \times R_{n-1} + R_n$$
となる。

【0098】ここで、R<sub>i</sub>は、0 ≤ R<sub>i</sub> ≤ i-1を満たす整数である。整数Xから、R<sub>2</sub>、R<sub>3</sub>、…、R<sub>(n-2)</sub>、R<sub>(n-1)</sub>、R<sub>n</sub>が一意的に決まるのは明らかである。また逆も明らかである。よって、0 ≤ X ≤ (n!-1)なる整数Xと、R<sub>2</sub>、R<sub>3</sub>、…、R<sub>(n-2)</sub>、R<sub>(n-1)</sub>、R<sub>n</sub>は1対1に対応していることが分かる。

【0099】・数列と入れ替えの対応について

先に述べたように、本実施の形態1においては、初期決定配列V1を、整数Xに従って各要素を入れ替えて配列Vを出力するわけである。先にも述べたように、その入れ替えは、数列R<sub>n</sub>、R<sub>(n-1)</sub>、R<sub>(n-2)</sub>、…、R<sub>3</sub>、R<sub>2</sub>により定まる。なぜならば、カウンタ値cがnである時の入れ替え処理終了時点での配列Vのn番目の要素は、初期決定配列V1の(R<sub>n</sub>+1)番目の要素である。以降は、配列Vのn番目の要素は入れ替わらない。同様に、カウンタ値cがiの時の入れ替え処理終了時点での配列Vのn番目の要素は、カウンタ値cが(i+1)の時の入れ替え処理終了時点での配列Vの(R<sub>i+1</sub>)番目の要素である。以降は、配列Vのi番目の要素は入れ替わらない。即ち、Vのn番目からi番目の要素までは変わらない。またカウンタ値cが(i+1)の時の入れ替え処理終了時点での配列Vは、数列R<sub>n</sub>、R<sub>(n-1)</sub>、…、R<sub>(i+1)</sub>により定まっているからである。

【0100】また、初期決定配列V1のn個の要素の入れ替え方は、「n個の元の置換」と考えられる。任意の「n個の元の置換」に対して、その置換は上に説明した配列要素置換部120の処理内容により実現できる。以下にその説明を行う。置換の表現として、入力であるn



個の順序付要素 $V[1], V[2], \dots, V[n]$ を置換した結果である順序列を用いることにする。例えば、 $(V[\sigma 1], V[\sigma 2], \dots, V[\sigma n])$ で1つの置換を表す。但し集合として $\{1, 2, \dots, n\} = \{\sigma 1, \sigma 2, \dots, \sigma n\}$ である。

【0101】この時、 $\sigma n$ の値は、上記 $R\_n$ の値により定まるとは明らかである。また $\sigma(n-1)$ の値も、 $R\_n, R_{(n-1)}$ の値により定まり、 $\sigma n$ の値とは異なる全ての値から一様に選ばれることは明らかである。以下同様にして、 $\sigma i$ の値は、 $R\_n, R_{(n-1)}, \dots, R_{(i+1)}$ の値により定まり、 $\sigma n, \sigma(n-1), \dots, \sigma(i+1)$ の値とは異なる全ての値から決まる。よって、上記配列要素置換部120の処理により、初期決定配列 $V1$ の全ての置換方法が決まっている。即ち、 $0 \leq X \leq (n! - 1)$ なる整数 $X$ と、初期決定配列 $V1$ の置換方法は1対1に対応していることが分かる。

【0102】今まで説明してきた議論は、初期決定配列 $V1$ の各要素の値が同じであるかどうかは関係なく、各要素の位置情報、即ちインデックスが違えば、違う要素と考えて行ってきた。しかし実際には、初期決定配列 $V1$ の各要素は、 $n1$ 個の1、 $n2$ 個の-1、 $(n-n1-n2)$ 個の0から構成されている。出力される配列 $V$ において、1が入っている $n1$ 個の要素の位置集合が同じであれば、配列 $V$ の値も同じである。このことは、 $n2$ 個の-1、 $(n-n1-n2)$ 個の0についても同様のことが言える。以下では、どれくらい同じ出力配列を出すことがあるかについて説明していく。

【0103】次に、配列要素置換部120からの出力の一意性について説明する。先に述べたように、 $0 \leq X \leq (n! - 1)$ を満たす整数 $X$ と「 $n$ 個の元の置換」は1対1に対応する。その変換を $\tau$ とする。配列要素置換部120は、整数 $X$ に対応する $\tau$ を用いて、 $\tau(V1)$ を $V$ として出力していることになる。 $\tau$ から $\tau(V1)$ は一意的に決定する。次にある $\tau 0$ に対して、

【0104】 $\tau 0(V1) = \tau 1(V1)$ となる変換 $\tau 1$ がいくつ存在するかについて説明する。 $\tau 0(V1)$ における $n1$ 個の1については、その位置が入れ替わっても、 $\tau 0(V1)$ の値としては変わらない。同様に、 $n2$ 個の-1についても、 $(n-n1-n2)$ の0についても、その位置が入れ替わっても $\tau 0(V1)$ の結果は変わらない。しかし、1入っている位置と0の入っている位置が入れ替わったり、1の入っている位置と-1の入っている位置が入れ替わったりすると、 $\tau 0(V1)$ の結果が変わってしまう。ゆえに、 $n1$ 個の1、 $n2$ 個の-1、 $(n-n1-n2)$ 個の0内の位置の入れ替えの組み合わせを分しか、取りうる $\tau 1$ は存在しない。1が $n1$ 個であるため、 $(n1)!$ 種類ある。同様に、-1が $n2$ 個、0が $(n-n1-n2)$ 個の入れ替えは、それぞれ、 $(n2)!, (n-n1-n2)!$ 種類

ある。従って、 $\tau 1$ は $(n1)! \times (n2)! \times (n-n1-n2)!$ 種類存在する。

【0105】ゆえに、実施の形態1の配列出力部100は、 $n!$ 種類の整数 $X$ を $n! / ((n1)! \times (n2)! \times (n-n1-n2)!)$ 種類の置換に変換可能であることがいえる。また、一意性については、先の議論が置換の種類に関係なく、 $n1, n2$ により決まっていることは明らかである。

【0106】以上説明したように、配列出力部100は、入力される整数 $X$ に基づいて一様に $n$ 次元の配列を出力することができる。また、以上の説明から配列出力部100は、同じ入力に対していつも同じ出力を行うことは明らかである。

【0107】以上は、 $X < n!$ の場合についての議論であるが、 $X > n!$ の場合も $X$ を $n!$ で割った余りを取ることによって、同様の議論が可能である。

【0108】実際のパラメータの場合について説明する。 $n=263, n1=16, n2=16$ とする時、 $L(n, n1, n2)$ に属する配列は、 $n! / ((n1)! \times (n2)! \times (n-n1-n2)!)$   $\approx 2^{163}$ 種類存在する。ここで、 $x^y$ は $x$ の $y$ 乗を表している。 $n! \approx 2^{1741}$ であり、ハッシュ関数の分布の一意性を保持するためには、ハッシュ関数の出力長として、1741ビット以上必要になる。出力が1741ビット以上のハッシュ関数は現在のところ存在しないが、ハッシュ関数を用いて、ハッシュ関数の出力長より長いハッシュ関数値を計算する方法がある。そのため、出力が1741ビット以上のような長いハッシュ関数値が必要であっても、安全な暗号方式の構成することを妨げるような問題にはならない。

【0109】このように本実施の形態によれば、配列出力部100は、入力された整数 $X$ に基づいて一様に $n$ 次元の配列を出力することができる。そのため、NTRU暗号にFOSRTを適用した場合において、配列出力部100は、ハッシュ関数部40より出力されるハッシュ関数値 $H(m)$ を基に、一様に $n$ 次元の配列を出力することができ、ハッシュ関数部40によるハッシュ関数の分布の一意性を保持することが可能となる。従って、暗号化装置10は、安全性の高い暗号文 $e(v)$ を生成することができる。

【0110】さらに、配列出力部100は、整数 $X$ のみに基づいて配列 $V$ を設定しているため、メモリーテーブルを用いる必要がなく、メモリーが少なくてもよいという効果が得られる。

【0111】また、図1(b)の復号化装置15の配列出力部105も配列出力部100と同様の構成をしており、暗号化装置10が暗号化した暗号文の復号が可能となる。

【0112】尚、本実施の形態では、暗号化装置10および復号化装置15を構成する各部が行う処理は、マイ



クロコンピュータのソフトウェアで行うものとしたが、電子回路やICなどのハードウェアで動作するようにしてもよい。

【0113】また、配列出力部100を暗号化装置10に用いて、ハッシュ関数値を基に配列を出力する構成としたがこれに限定されない。

【0114】さらに、図2に示した配列出力部100は、初期配列決定部110と配列要素置換部120とを備え、初期配列決定部110が決定した初期決定配列V1の各要素を配列要素置換部120が整数Xを基に置換する構成であるが、整数Xと予め設定された初期決定配列V1を入力とし、その初期決定配列V1を、整数Xを基に置換して配列Vを出力する配列要素置換部120と同様の処理を行う構成の配列出力部（以下、配列出力部100aという）であってもよい。

【0115】このような構成の配列出力部100aは、配列要素置換部120と同様の処理を行うため、ハッシュ関数の分布の一意性を保持することができ、さらにテーブルを用いず、整数Xのみの情報より配列V1を置換しているため、メモリが少なくてもよいという効果が得られる。

【0116】また、配列出力部100aは、整数Xを鍵、初期決定配列V1をメッセージ、配列Vを暗号文とする暗号装置または、暗号方法であってもよい。また、配列出力部100aを使用する暗号装置または、暗号方法であってもよい。

【0117】（実施の形態2）以下、本発明における第2の実施の形態に係る暗号化装置について説明する。本実施の形態における暗号化装置は、図1(a)の暗号化装置10と比べて配列出力部100が異なる構成の配列出力部200となっている。その他の構成は共通しているため、その説明を省略する。

【0118】図5は、本実施の形態における配列出力部200の構成を示すブロック図である。この配列出力部200は、整数Xを入力とし、 $L(n, n1, n2)$ に属する配列V20を出力するものである。ここで、 $L(n, n1, n2)$ は、 $n1$ 個の要素が1、 $n2$ 個の要素が-1、その他の $(n - n1 - n2)$ 個の要素が0である $n$ 次元配列全体であり、 $n, n1, n2$ は配列出力部200に予め設定されているものとする。

【0119】配列出力部200は、第1の数配置部210と第2の数配置部220からなり、配列出力部100と同様にマイクロコンピュータのソフトウェア、あるいは電子回路などのハードウェアにより処理を実行する。

【0120】第1の数配置部210は、整数Xを入力とし、 $n1$ 個の要素が1、その他の要素が0である $n$ 次元の配列V10と、整数Xに所定の演算が行われた整数X1を第2の数配置部220に出力する。第1の数配置部210は、すべての配列要素が0の配列要素を暫定的に決定し、0の配列要素を整数Xに基づいて1に改編す

る。

【0121】第2の数配置部220は、第1の数配置部210より出力された配列V10と、整数X1とを入力とし、 $n1$ 個の要素が1、 $n2$ 個の要素が-1、その他の $(n - n1 - n2)$ 個の要素が0である $n$ 次元の配列V20を出力する。ここで、第2の数配置部220は、第1の数配置部210が出力する配列の0の配列要素を-1に改編する。

【0122】まず、第1の数配置部210の動作を説明する。図6は、第1の数配置部210が行う処理を示すフローチャートである。第1の数配置部210は、以下のようなステップの処理で行われる。なお、以下では、配列V10の（左から） $i$ 番目の要素を $V10[i]$ とする。また、カウンタ $c1'$ の値をカウンタ値 $c1$ 、カウンタ $c2'$ の値をカウンタ値 $c2$ とする。

【0123】図7は第1の数配置部210における配列V10の各段階での配列状態を示している。まず、第1の数配置部210は、整数Xを変数Y1に代入する（ステップS201）。

【0124】次に、第1の数配置部210は、配列V10を全ての要素を0（整数P1）とする（ステップS202）。ここで、初期配列が決定される。次に、第1の数配置部210は、カウンタ $c1'$ のカウンタ値 $c1$ を1に設定する（ステップS203）。

【0125】次に、第1の数配置部210は、カウンタ $c2'$ のカウンタ値 $c2$ を $n$ に設定する（ステップS204）。次に、第1の数配置部210は、変数Y1（被除数）をカウンタ値 $c2$ （除数）で割った商Sと余りRを求める（ステップS205）。

【0126】次に、第1の数配置部210は、配列V10の0である要素の中で、左から $(R+1)$ 番目の要素を1（整数P2）に設定する（ステップS206）。次に、第1の数配置部210は、商Sを変数Y1に代入する（ステップS207）。

【0127】次に、第1の数配置部210は、カウンタ値 $c1 = n1$ であるかを判定し（ステップS208）、カウンタ値 $c1 = n1$ でないと判定した場合は（ステップS208のNo）、配列V10の1である要素が $n1$ 個に達していないとして、カウンタ値 $c1$ をカウンタアップし（ $c1 \leftarrow c1 + 1$ ）、カウンタ値 $c2$ をカウンタダウン（ $c2 \leftarrow c2 - 1$ ）する処理に移る（ステップS209）。そして、第1の数配置部210は、再度、変数Y1をカウンタ値 $c2$ で割った商Sと余りRを求める処理（ステップS205）を行う。

【0128】一方、第1の数配置部210は、カウンタ値 $c1 = n1$ であると判定した場合は（ステップS208のYes）、配列V10の1である要素が $n1$ 個になったとして、配列V10と、変数Y1（整数X1）の値を第2の数配置部220に出力して（ステップS210）、処理を終了する。



【0129】このように、第1の数配置部210は、カウンタc1'のカウンタ値c1がn1となるまで、商Sと余りRを求める処理（ステップS205）からカウンタ値c1をカウントアップし、カウンタ値c2をカウントダウンする処理（ステップS209）までを繰り返す。そして、カウンタc1'のカウンタ値c1がn1となると、すなわち配列V10の要素のうち1がn1個となると、第1の数配置部210は、配列V10を第2の数配置部220に出力する。

【0130】この第1の数配置部210の動作を、具体例を用いて説明する。実際に入力される整数 $X=5644$ とし、また第1の数配置部210が出力するn次元の配列V10が、例えば8次元の配列であり（ $n=8$ ）、3個（ $n1=3$ ）の要素が1、その他の5個の要素が0である場合の例を用いて説明する。

【0131】まず、第1の数配置部210は、変数Y1に5644を代入する（ステップS201）。次に、第1の数配置部210は、配列V10の配列状態を、図7の配列V11に示す、すべての要素が0である配列状態とする（ステップS202）。

【0132】次に、第1の数配置部210は、カウンタ値c1=1とし（ステップS203）、カウンタ値c2=8（ステップS204）とする。次に、第1の数配置部210は、変数Y1=5644、およびカウンタ値c2=8より、商 $S=705$ 、余り $R=4$ を求める（ステップS205）。

【0133】次に、第1の数配置部210は、配列V10の0である要素のうち左から（ $R+1$ ）番目、すなわち5番目の要素V10[5]を1に設定し、図7の配列V12に示す配列状態とする（ステップS206）。次に、第1の数配置部210は、変数Y1に705（=商S）を代入する（ステップS207）。

【0134】そして、カウンタ値c1=1であり、カウンタ値c1=3（=n1）でない（ステップS208のNo）、第1の数配置部210は、カウンタ値c1=2（カウントアップ）、カウンタ値c2=7（カウントダウン）とする（ステップS209）。

【0135】次に、第1の数配置部210は、変数Y1=705、およびカウンタ値c2=7より、再度、商S、余りRを求める（ステップS205）。商 $S=100$ 、余り $R=5$ となる。

【0136】次に、第1の数配置部210は、配列V10の0である要素のうち左から6番目の要素を1に設定する。この設定前の配列V10の配列状態は、図7の配列V12に示す配列状態である。配列V12の0の要素は、V12[5]以外の要素である。配列V12の0の要素の中で6番目の要素は、V12[7]であるので、V12[7]の0を1にする（ステップS206）。これにより、図7に示す配列V13となる。

【0137】このように、第1の数配置部210は、配

列V10の要素のうち1がn1個（本例では $n1=3$ ）となるまで、余りRに応じて配列V10の0の要素を1にし、図7の配列V14に示すように1である要素が3個となれば、その配列を配列V10として第2の数配置部220に出力する。

【0138】次に、第2の数配置部220の動作を説明する。図8は、第2の数配置部220が行う処理を示すフローチャートである。図9は第2の数配置部220における配列V20の各段階での配列状態を示している。なお、以下では、配列V20の（左から）i番目の要素をV10[i]とする。また、カウンタc1'の値をカウンタ値c1、カウンタc2'の値をカウンタ値c2とする。

【0139】まず、第2の数配置部220は、第1の数配置部210から出力された変数Y1（整数X1）の値を変数Y2に代入する（ステップS301）。次に、第2の数配置部220は、第1の数配置部210から出力された配列V10を配列V20に代入する（ステップS302）。次に、第2の数配置部220は、カウンタ値c1を1に設定する（ステップS303）。

【0140】次に、第2の数配置部220は、カウンタ値c2を（ $n-n1$ ）に設定する（ステップS304）。次に、第2の数配置部220は、変数Y2（被除数）をカウンタ値c2（除数）で割った商Sと余りRを求める（ステップS305）。次に、第2の数配置部220は、配列V20の0である要素の中で、左から（ $R+1$ ）番目の要素を-1に設定する（ステップS306）。次に、第2の数配置部220は、商Sを変数Y2に代入する（ステップS307）。

【0141】次に、第2の数配置部220は、カウンタ値c1=n2であるかを判定し（ステップS308）、カウンタ値c1=n2でないと判定した場合は（ステップS308のNo）、配列V20の-1である要素がn2個に達していないとして、カウンタ値c1をカウントアップし（ $c1 \leftarrow c1+1$ ）、カウンタ値c2をカウントダウン（ $c2 \leftarrow c2-1$ ）する処理（ステップS309）に移る。つづいて、第2の数配置部220は、再度、変数Y2をカウンタ値c2で割った商Sと余りRを求める処理を行う（ステップS305）。

【0142】一方、第2の数配置部220は、カウンタ値c1=n2であると判定した場合は（ステップS308のYes）、配列V20の-1である要素がn2個になったとして、配列V20を出力して（ステップS310）、処理を終了する。

【0143】このように、第2の数配置部220は、カウンタ値c1=n2となるまで、商Sと余りRを求める処理（ステップS305）からカウンタ値c1をカウントアップし、カウンタ値c2をカウントダウンする処理（ステップS309）までを繰り返し、カウンタ値c1=n2となると、配列V20の要素のうち-1がn2個



(本例では $n2=2$ )となったとして、配列V20を出力する。

【0144】この第2の数配置部220の動作を、具体例を用いて説明する。実際に第1の数配置部210から出力される整数 $X1=150$ とし、また第2の数配置部220が出力する $n$ 次元の配列V20が、例えば8次元の配列であり( $n=8$ )、3個( $n1=3$ )の要素が1であり、2個( $n2=2$ )の要素が-1であり、その他の3個の要素が0である例を用いて説明する

【0145】まず、第2の数配置部220は、変数Y2に150を代入する(ステップS301)。次に、第2の数配置部220は、配列V20に、第1の数配置部210から出力される3個の要素が1であり、その他の5個の要素が0である配列V10を代入する。図9の配列V21がその代入した配列である(ステップS302)。

【0146】次に、第2の数配置部220は、カウント値 $c1=1$ とし(ステップS303)、カウント値 $c2=n-n1=8-3=5$ とする(ステップS304)。次に、第2の数配置部220は、変数Y2=150、およびカウント値 $c2=5$ より、商 $S=30$ 、余り $R=0$ を求める(ステップS305)。

【0147】次に、第2の数配置部220は、配列V20の0である要素のうち左から1番目の要素、すなわちV20[1]を-1にする(ステップS306)。図9の配列V22がその配列状態を示している。次に、第2の数配置部220は、変数Y2=30(=商S)とする(ステップS307)。

【0148】そして、カウント値 $c1=1$ であり、カウント値 $c1$ は $n2(=2)$ でない(ステップS308のNo)、第2の数配置部220は、カウント値 $c1=2$ (カウントアップ)、カウント値 $c2=4$ (カウントダウン)とする(ステップS309)。

【0149】次に、第2の数配置部220は、変数Y2=30、およびカウント値 $c2=4$ より、再度、商S、余りRを求める(ステップS305)。商 $S=7$ 、余り $R=2$ となる。

【0150】次に、第2の数配置部220は、配列V20の0である要素のうち左から $(R+1)$ 番目、すなわち3番目の要素を-1に設定する。設定前の配列V20の配列状態は、図9に示す配列V22であり、0の要素は、V22[3]、V22[4]、V22[6]、V22[8]である。配列V22の0の要素の中で左から3番目の要素は、V22[6]であるので、V22[6]の0を-1にする(ステップS306)。この設定後の配列状態は、図9に示す配列V23である。

【0151】次に、第2の数配置部220は、変数Y2に7(=商S)を代入する(ステップS307)。

【0152】そして現時点で、カウント値 $c1=2$ 、即ちカウント値 $c1=n2$ であるため(ステップS308

のYes)、第2の数配置部220は、配列V20を出力する(ステップS310)。この出力される配列20は、図9の配列V23に示す配列状態である。

【0153】このように、配列出力部200は、ハッシュ関数値 $H(m)$ である整数Xから $n1$ 個の要素が1、 $n2$ 個の要素が-1、 $(n-n1-n2)$ 個の要素が0である $n$ 次元の配列を出力する。

【0154】上述した配列出力部200は、 $0 \leq X \leq ((n!)/(n-n1-n2)!)-1$ を満たす整数Xに基づいて、一様な配列 $L(n, n1, n2)$ を出力している。以下では、 $0 \leq X \leq ((n!)/(n-n1-n2)!)-1$ を満たす整数Xに限定して説明を行う。

【0155】第1の数配置部210では、整数Xに基づいた位置の配列V10の要素を1に設定している。以下では、この要素を1に設定する処理について具体的に見ていく。また、実施の形態1と同様に、第1の数配置部210の構成において、カウント値 $c2$ におけるステップS205の余りを $R\_c2$ とする。全体の説明の流れは、以下の通りである。

- (1) 入れ替え処理が重複しないことについて説明する。
- (2) (1)の結果を利用して、数Xと配列要素置換部における処理内容が1対1に対応していることを説明する。
- (3) 同じ配列を出力する数Xが何種類あるかを計算する。

【0156】この(3)の計算において配列に依らず、整数Xの種類の数が同じであることが示すことが可能であるため、入力に対して出力が一様に分布することがいえる。

【0157】まず、配置処理が重複しないことについて説明する。以下では、まず、配列の同じ位置に対する値(1または-1)の配置が複数回起こらない、即ち、配置が終わった位置は、その処理以降の配置の対象にならないことについて説明する。

【0158】・カウント値 $c2$ が $n$ の時  
ステップS206では、配列V10の0である要素の中で、 $R\_n+1$ 番目の要素を1に設定している。ここで、カウント値 $c2$ が $n$ の時は、 $R\_n+1$ 番目の要素を1に設定する前のV10は全ての要素が0であるため、単純に、 $R\_n+1$ 番目を1に設定していることになる。

【0159】・カウント値 $c2$ が $n-1$ の時  
ステップS206では、配列V10の0である要素の中で、 $R\_n+1$ 番目の要素を1に設定している。ここで、配列V10の0である要素を1に設定する要素の対象としているため、カウント値 $c2$ が $n$ の時に設定した $R\_n$ 番目の要素はその対象とならないことに注意する。配列V10が0である要素の中で、 $R\_n$

10

20

30

40

50



$-1) + 1$  番目の要素は、 $R_{-}(n-1) < R_{-}n$  である時は、単純に、配列V10の $R_{-}(n-1) + 1$  番目の要素になっている。 $R_{-}(n-1) > R_{-}n$  である時は、配列V10の $R_{-}(n-1) + 2$  番目の要素になっている。

【0160】上記のように、カウント値c2がiの時に1に設定した要素は、カウント値c2がi+1以降のステップで、1に設定する要素の対象にならないため、1回設定した要素をさらに設定することがない。

【0161】第1の数配置部210の処理が終了した後

従って、カウント値c2がnから $(n-n1+1)$ までの全ての処理が終わった時点で、配列V10は、n1個の要素が1であり、その他の $(n-n1)$ 個の要素が0になっている。上記の第1の数配置部210の議論は、第2の数配置部220でも同様に行える。第2の数配置部220の処理が終了した時点で、配列V20は、n1個の要素が1であり、n2個の要素が-1であり、その他の $(n-n1-n2)$ 個の要素が0になっている。

【0162】第1の数配置部210と第2の数配置部の処理の一意性について説明する。以下では、上記の結果（値の配置が重複しない）を利用して、整数Xと数配置部における処理内容が1対1に対応していることを説明していく。

【0163】・整数Xと数列の対応

整数Xを $0 \leq X \leq ((n!) / (n-n1-n2)!) - 1$  を満たす整数とする。この時、実施の形態1と同様に、カウント値c2がiの時の第1の数配置部のステップS205のRを $R_{-}i$ 、第2の数配置部の同様の処理を行うステップS305のRも $R_{-}i$ とする。ここで、第1の数配置部210のカウント値c2は、nから $n-n1+1$ までの範囲にあり、第2の数配置部220のカウント値c2は、 $n-n1$ から $n-n1-n2+1$ までの範囲にあるため、カウント値c2が第1の数配置部と、第2の数配置部の間で重なることがないことを注意しておく。この時、整数Xは、実施の形態1と同様に以下のようにおける。

【0164】 $X = (n \times (n-1) \times \dots \times (n-n1-n2+2)) \times R_{-}(n-n1-n2+1) + (n \times (n-1) \times \dots \times (n-n1-n2+3)) \times R_{-}(n-n1-n2+2) \dots + (n \times (n-1)) \times R_{-}(n-2) + n \times R_{-}(n-1) + R_{-}n$

従って、整数Xは、数列 $R_{-}(n-n1-n2+1)$ 、 $\dots$ 、 $R_{-}n$ が1対1対応していることが分かる。

【0165】・数列と整数配置

$R_{-}i$  ( $n-n1-n2+1 \leq i \leq n$ ) によって、カウンタc2がiの時に、配列V20の $R'_{-}i+1$  番目が $n-n1+1 \leq i \leq n$ の場合は、1、 $n-n1-n2+1 \leq i \leq n-n1$ の場合は、-1に設定されたと考えられる。その時、数列 $R_{-}(n-n1-n2+1)$ 、 $\dots$ 、 $R_{-}n$

$-n$ と、上記の設定は、順序即ち、1または、-1に設定された時のカウンタc2の値も含めて考えると1対1対応することが分かる。

【0166】カウント値c2がj ( $j \neq i$ ) の時に、 $R'_{-}i+1$  番目の要素をy (yは1または、-1) に設定し、カウント値c2がiの時に $R'_{-}j+1$  番目の要素をyに設定するように、yに設定する時のカウント値c2の値が入れ替わっても、同じ配列V20を出力する。この他に同じ配列Vを出力するようなケースは存在しない。このようなカウント値c2の値の入れ替えは、実施の形態1と同様に考えると、yが1の場合で、 $(n1)!$ 個、yが-1の場合で、 $(n2)!$ 個存在する。従って、同じ配列V20を出力する整数Xは、 $(n1)! \times (n2)!$ 種類存在することになる。

【0167】 $L(n, n1, n2)$  に属する配列は、 $n! / ((n1)! \times (n2)! \times (n-n1-n2)!)$ 種類存在する。ゆえに、実施の形態2の配列出力部200は、 $n! / ((n1)! \times (n2)! \times (n-n1-n2)!)$ 種類の配列を一樣に出力可能であることがいえる。

【0168】また、以上の説明から配列出力部200は、同じ入力に対していつも同じ出力を行うことは明らかである。

【0169】以上は、 $X < n! / (n-n1-n2)!$  の場合についての議論であるが、 $X > n! / (n-n1-n2)!$  の場合もXを $(n! / (n-n1-n2)!)$ で割った余りを取ることによって、同様の議論が可能である。

【0170】ここで、実施の形態2と実施の形態1の効果との差異を説明する。実施の形態1では、カウンタc'のカウント値cは、nから2までの範囲で動く。それに対して、実施の形態2では、第1の数配置部210と第2の数配置部220において、カウンタc2'のカウント値c2は、nから $(n-n1-n2+1)$ までの範囲で動く。そのため、整数Xの値が、実施の形態1では、 $n!$ 種類存在するのに対し、実施の形態2では、 $(n! / (n-n1-n2)!)$ 種類存在する。ゆえに、実施の形態2は実施の形態1より、入力の種類が $(1 / (n-n1-n2)!)$ に削減でき、必要な入力のビット長も短くてすむ。

【0171】実際のパラメータの場合について説明する。 $n=263$ 、 $n1=16$ 、 $n2=16$ とする時、 $L(n, n1, n2)$  に属する配列は、 $n! / ((n1)! \times (n2)! \times (n-n1-n2)!) \approx 2^{163}$  である。この時、 $n! / (n-n1-n2)! \approx 2^{55}$  であり、ハッシュ関数の分布の一樣性を保持するためには、ハッシュ関数の出力長として、255ビット以上必要になる。実施の形態1に比べると必要なハッシュ関数出力長が1486ビット小さく、実施の形態1より



効率がよい。

【0172】このように、本実施の形態によれば、配列出力部200は、整数Xに基づいてn次元の配列を一樣に出力することができる。そのため、NTRU暗号にFOSRTを適用した場合において、図1(a)の暗号化装置10の配列出力部100の代わりにこの配列出力部200とし、ハッシュ関数部40より出力されるハッシュ関数値H(m)に基づいて、配列出力部200がn次元の配列を一樣に出力することで、ハッシュ関数の分布の一樣性を保持することが可能となり、暗号化装置10

で生成する暗号文の安全性を高めることができる。  
【0173】さらに、配列出力部200は、整数Xのみの情報より配列V20を設定しているため、メモリテーブルを用いる必要がなく、メモリが少なくてもよいという効果が得られる。

【0174】また、図1(b)の復号化装置15の配列出力部105の代わりに、この配列出力部200を用いることで暗号化された暗号文の復号が可能となる。

【0175】尚、また、配列出力部200を暗号化装置10に用いて、ハッシュ関数値を基に配列を出力する構成としたがこれに限定されない。

【0176】(実施の形態3) 本発明における第3の実施の形態に係る暗号化装置を説明する。本実施の形態の暗号化装置は、図1の暗号化装置10と比べて配列出力部100が異なる構成の配列出力部300となっている。その他の構成は共通しているため、その説明を省略する。

【0177】 本実施の形態における配列出力部300を、図面を用いて説明する。図10は、本実施の形態における配列出力部300の構成を示すブロック図である。

【0178】この配列出力部300は、整数Xを入力とし、L(n, n1, n2)に属する配列V40を出力するものである。ここで、n, n1, n2は予め設定され、この配列出力部300に与えられているものとする。

【0179】配列出力部300は、第1の数配置部310と第2の数配置部320からなり、配列出力部100と同様、マイクロコンピュータのソフトウェア、あるいは電子回路などのハードウェアにより処理を実行する。

【0180】第1の数配置部310は、整数Xを入力とし、n1個の要素が1、その他の要素が0であるn次元の配列V30と、整数Xに所定の演算が行われた整数X2を出力する。第1の数配置部310は、すべての配列要素が0の配列要素を暫定的に決定し、0の配列要素を整数Xに基づいて1に改編する。

【0181】第2の数配置部320は、第1の数配置部310より出力された配列V30および整数X2を入力として、n1個の要素が1、n2個の要素が-1、その他の(n-n1-n2)個の要素が0であるn次元の配

列V40を出力する。ここで、第2の数配置部320は、第1の数配置部310が出力する配列の0の配列要素を-1に改編する。

【0182】第1の数配置部310の動作を説明する。図11は、第1の数配置部310の処理を示すフローチャートである。第1の数配置部310は、以下のようなステップの処理で行われる。なお、以下では、配列V30の(左から)i番目の要素をV30[i]とする。また、カウンタc1'の値をカウンタ値c1、カウンタc2'の値をカウンタ値c2とする。また、C(s, t)は、s個のものからt個を選ぶ組み合わせの数を示す。具体的には、 $C(s, t) = s! / ((s-t)! \times t!)$ である。

【0183】まず、第1の数配置部310は、整数Xを変数Z1に代入し、配列V30を全ての要素が0のn次元配列とする(ステップS401)。ここで、初期配列が決定される。

【0184】次に、第1の数配置部310は、カウンタc1'のカウンタ値c1をn1に設定し、カウンタc2'のカウンタ値c2を1に設定する(ステップS402)。

【0185】次に、第1の数配置部310は、変数Z1  $\geq C(n-c2, c1)$  であるかを判定する(ステップS403)。第1の数配置部310は、変数Z1  $\geq C(n-c2, c1)$  であると判定した時(ステップS403のYes)、変数Z1にZ1-C(n-c2, c1)を代入し、カウンタc1'のカウンタ値c1をカウンタダウンし(c1←c1-1)、配列V30の(n-c2+1)番目の要素を1にする(V30[n-c2+1]←1)(ステップS404)。そして、第1の数配置部310は、カウンタc2'のカウンタ値c2をカウンタアップする(ステップS406)。

【0186】一方、第1の数配置部310は、変数Z1  $\geq C(n-c2, c1)$  でないと判定した時(ステップS403のNo)、配列V30の(n-c2+1)番目の要素を0にする(V30[n-c2+1]←0)(ステップS405)。そして、第1の数配置部310は、カウンタc2'のカウンタ値c2をカウンタアップする(ステップS406)。

【0187】このように、変数Z1とC(n-c2, c1)の大小関係に応じて、配列V30の(n-c2+1)番目の要素を0か1に設定する。そして、第1の数配置部310は、カウンタ値c2をカウンタアップすると(ステップS406)、カウンタ値c2>nを判定する(ステップS407)。

【0188】第1の数配置部310は、カウンタ値c2>nでないと判定した場合は(ステップS407のNo)、再度、変数Z1  $\geq C(n-c2, c1)$  であるかを判定する判定処理を行い(ステップS403)、その判定処理(ステップS403)からカウンタ値c2>n



を判定する(ステップS407)までの処理を、カウン  
ト値 $c2 > n$ となるまで繰り返す。

【0189】一方、第1の数配置部310は、カウン  
タ $c2 > n$ と判定した場合(ステップS407のYe  
s)、配列V30と、整数X2(整数 $X2 = \text{整数}X / C$   
( $n, n1$ ))とを第2の数配置部320に出力する。

【0190】この第1の数配置部310の動作を、具  
体例を用いて説明する。実際に入力される整数 $X = 50$ と  
し、また第1の数配置部310が出力する配列V30  
が、例えば8次元の配列であり( $n = 8$ )、4個( $n1 = 4$ )の要素が1、その他の4個の要素が0である場合  
の例を用いて説明する。

【0191】図13(a)は、配列V30の各段階の配  
列状態を示している。

【0192】まず、第1の数配置部310は、変数Z1  
に50を代入する(ステップS401)。

【0193】次に、第1の数配置部310は、カウン  
タ $c1'$ のカウンタ値 $c1$ を4とし、カウンタ $c2'$ のカ  
ウンタ値 $c2$ を1とする(ステップS402)。

【0194】この場合、変数 $Z1 (= 50) \geq C(7, 20$   
 $4)(= 35)$ であるので(ステップS403のYe  
s)、第1の数配置部310は、変数 $Z1 \leftarrow 50 - 35$   
 $= 15$ 、カウンタ値 $c1 = 3$ 、配列V30の左から8番  
目の要素V30[8] = 1とする(ステップS40  
4)。このときの配列V30の配列状態は図13に示す  
配列V31である。

【0195】次に、第1の数配置部310は、カウン  
タ値 $c2$ を2にカウントアップする(ステップS40  
6)。ここで、カウンタ値 $c2 > 8$ でないため(ステッ  
プS407のNo)、第1の数配置部310は配列V30  
の出力を行わず、再度、変数Z1と $C(n - c2, c$   
 $1)$ の大小関係を判定する(ステップS403)。

【0196】この場合、変数 $Z1 (= 15) < C(6,$   
 $3)(= 20)$ であるので(ステップS403のN  
o)、第1の数配置部310は、配列V30の左から7  
番目の要素V30[7] = 0とする(ステップS40  
5)。このときの配列V30の配列状態は図13に示す  
配列V32である。

【0197】このように、変数Z1と $C(n - c2, c$   
 $1)$ の大小関係に応じて、配列V30の各要素に順番に  
1か0を設定していき、全要素について決定したとき  
に、第1の数配置部310は、第2の数配置部320に  
配列V30を出力する。

【0198】上記の処理は、Schalkvijkのアル  
ゴリズムと呼ばれる。Schalkvijkのアル  
ゴリズムについて、Schalkvijk, "An Al  
gorithm for Source Coding", IT72-18, 1972が詳しい。以下では、  
この文献を「非特許文献4」とよぶ。

【0199】次に、第2の数配置部320の動作を説明 50

する。図12は、第2の数配置部320の処理を示すフ  
ローチャートである。第2の数配置部320は、以下の  
ようなステップの処理で行われる。

【0200】なお、以下では、配列V40の(左から)  
i番目の要素を $V40[i]$ とする。また、カウンタ $c$   
 $1'$ の値をカウンタ値 $c1$ 、カウンタ $c2'$ の値をカ  
ウンタ値 $c2$ とする。

【0201】まず、第2の数配置部320は、第1の数  
配置部310から出力された整数X2を変数Z2に代入  
し、配列V30を配列V40に代入し、さらに配列Wを  
全ての要素が0である( $n - n1$ )次元の配列とする  
(ステップS501)。

【0202】次に、第2の数配置部320は、カウン  
タ $c1'$ のカウンタ値 $c1$ を $n2$ 、カウンタ $c2'$ のカ  
ウンタ値 $c2$ を1とする(ステップS502)。

【0203】次に、第2の数配置部320は、変数Z2  
 $\geq C(n - n1 - c2, c1)$ であるかを判定し(ステ  
ップS503)、 $Z2 \geq C(n - n1 - c2, c1)$ で  
あると判定した時(ステップS503のYes)、変数Z  
2に $Z2 - C(n - n1 - c2, c1)$ を代入し、カ  
ウンタ値 $c1$ をカウンタダウンし( $c1 \leftarrow c1 - 1$ )、配  
列Wの( $n - n1 - c2 + 1$ )番目の要素を-1にする  
(ステップS504)。そして、第2の数配置部320  
は、カウンタ値 $c2$ をカウンタアップする(ステップS  
506)。

【0204】一方、第2の数配置部320は、 $Z2 \geq C$   
( $n - n1 - c2, c1$ )でないと判定した時(ステッ  
プS503のNo)、配列Wの( $n - n1 - c2 + 1$ )番  
目の要素を0とする(ステップS505)。そして、第  
2の数配置部320は、カウンタ値 $c2$ をカウンタア  
ップする(ステップS506)。

【0205】このように、変数Z2と $C(n - n1 - c$   
 $2, c1)$ の大小関係に応じて、配列Wの( $n - n1 -$   
 $c2 + 1$ )番目の要素が0か-1に設定される。

【0206】そして第2の数配置部320は、カウン  
タ値 $c2$ のカウントアップを行うと(ステップS50  
6)、カウンタ値 $c2$ が $c2 > n - n1$ であるかを判定  
する(ステップS507)。第2の数配置部320は、  
 $c2 > n - n1$ でないと判定した時(ステップS507  
のNo)、変数 $Z2 \geq C(n - n1 - c2, c1)$ であ  
るかを再度判定する判定処理を行い(ステップS50  
3)、 $c2 > n - n1$ となるまで、上記判定処理(ステ  
ップS503)から $c2 > n - n1$ の処理(ステップS  
507)を繰り返す。

【0207】すなわち、第2の数配置部320は、カ  
ウンタ値 $c2$ が1から( $n - n1$ )の各場合において、変  
数Z2と $C(n - n1 - c2, c1)$ の大小関係に応じ  
て、配列Wの( $n - n1 - c2 + 1$ )番目の要素が0か  
-1に設定される。

【0208】一方、第2の数配置部320は、 $c2 > n$



$-n1$ であると判定すると(ステップS507のYes)、カウント値 $c1$ を1、カウント値 $c2$ を1とする処理に移る(ステップS508)。

【0209】次に、第2の数配置部320は、配列V40の $c1$ 番目の要素V40[ $c1$ ]が1であるかの判定処理を行い(ステップS509)、V40[ $c1$ ]=1と判定したとき(ステップS509のYes)、カウント値 $c1$ をカウントアップし( $c1 \leftarrow c1 + 1$ ) (ステップS512)、再度、上記判定処理(ステップS509)を行う。

【0210】一方、第2の数配置部320は、V40[ $c1$ ]=1でないと判定したとき(ステップS509のNo)、配列V40の $c1$ 番目の要素V40[ $c1$ ]に配列Wの $c2$ 番目の要素W[ $c2$ ]を代入し、カウント値 $c2$ をカウントアップ( $c2 \leftarrow c2 + 1$ )する(ステップS510)。

【0211】次に、第2の数配置部320は、カウント値 $c2$ が $c2 > n - n1$ であるかを判定し(ステップS511)、 $c2 > n - n1$ でないと判定したとき(ステップS512のNo)、カウント値 $c1$ をカウントアップする処理( $c1 \leftarrow c1 + 1$ ) (ステップS512)に移る。

【0212】一方、第2の数配置部320は、カウント値 $c2$ が $c2 > n - n1$ であると判定した時は、配列V40を外部に出力して処理を終了する。

【0213】ここで、上述したカウント値 $c1$ を1、カウント値 $c2$ を1とする処理(ステップS508)から配列V40を出力する(ステップS512まで)までの処理では、V[ $c1$ ]の0の要素にW[ $c2$ ]の要素を順番に代入している。

【0214】この第2の数配置部320の動作を、具体例を用いて説明する。実際に入力される整数 $X2 = 20$ とし、また第1の数配置部320が出力する配列V40が、例えば8次元の配列であり( $n = 8$ )、4個( $n1 = 4$ )の要素が1、2個( $n2 = 2$ )の要素が-1、その他の2個の要素が0である場合の例を用いて説明する。

【0215】図13(b)は、配列V40の各段階の配列状態を示しており、図13(c)は、配列Wの各段階の配列状態を示している。

【0216】まず、第2の数配置部320は、変数 $Z2 = 20$ とし、配列V40に例えば、第1の数配置部310から出力された図13(b)の配列V41に示す配列状態を代入し、さらに配列Wをすべての要素が0の( $n - n1$ )次元(4次元)の配列とする(ステップS501)。

【0217】次に、第2の数配置部320は、カウンタ $c1'$ のカウント値 $c1$ を2に設定し、カウンタ $c2$ を1に設定する(ステップS502)。

【0218】次に、第2の数配置部320は、変数 $Z2$

と $C(n - n1 - c2, c1)$ の大小関係を判定し(ステップS503)、変数 $Z2 (= 20) \geq C(3, 2) (= 3)$ であるため(ステップS503のYes)、第2の数配置部320は、変数 $Z2 = 20 - 3 = 17$ とし、カウント値 $c1$ を1とし、配列Wの左から4番目の要素、すなわちW[ $n - n1 - c2 + 1$ ]を-1とする(ステップS504)。この配列状態は図13(c)の配列W1に示した通りである。

【0219】このように、配列Wの各要素に0か-1を設定する。図13(c)の配列W2は、全要素の設定された配列状態の一例である。

【0220】そして、第2の数配置部320は、配列Wの全要素を設定すると(ステップS507のYes)、カウンタ $c1'$ のカウント値 $c1$ を1にし、カウンタ $c2'$ のカウント値 $c2$ を1にする(ステップS508)。この以降の処理で、配列V41の0の要素に配列Wの各要素を代入する。

【0221】現時点での配列V40の配列状態を配列V41とし、配列Wの配列状態を配列W2とする。まず、配列V41の $c1$ 番目の要素、すなわちV41[1]=0であるため(ステップS509のNo)、V41[1]に配列Wの $c2$ 番目の要素、すなわちW[1]である0を代入する。図13(b)の配列V42がその配列状態である。

【0222】このように、第2の数配置部320は、配列V41の0の要素に配列W2の全要素を代入し(ステップS509からステップS512の処理)、配列V41を出力する。図13(b)の配列V43が配列V41の0の要素に配列W2の要素を代入した配列状態である。

【0223】ここで、配列出力部300全体の動作を説明する。まず、第1の数配置部310は、整数 $X$ を入力とし、 $n1$ 個の要素が1、他の要素が0である $n$ 次元の配列V30と、整数 $X2$ (整数 $X2 = \text{整数}X / C(n, n1)$ )とを第2の数配置部320に出力する。

【0224】次に、第2の数配置部320は、第1の数配置部310から出力された配列V30、整数 $X2$ を入力とし、 $n1$ 個の要素が1、 $n2$ 個の要素が-1、他の要素が0である $n$ 次元の配列V40を出力する。

【0225】本実施の形態では、Schalkvijkのアルゴリズムを応用している。具体的には、第1の数配置部310での配列V30における1の要素の配置場所を決定する箇所と、第2の数配置部320での配列V40における-1の要素の配置場所を決定する箇所に、Schalkvijkのアルゴリズムを使用している。

【0226】この配列出力部300は、 $0 \leq X \leq C(n, n1) \times C(n - n1, n2) (= n! / ((n1)! \times (n2)! \times (n - n1 - n2)!))$ を満たす整数 $X$ を $L(n, n1, n2)$ に1対1に変換している。

【0227】以下では、 $0 \leq X \leq C(n, n1) \times C$



( $n-n1, n2$ )を満たす整数Xに限定して、1対1に変換していることについて説明を行う。

【0228】第1の数配置部310、第2の数配置部320では、Schalkvijkアルゴリズムを利用して、1、-1の配置場所を決定している。Schalkvijkアルゴリズムは、入力の出力の種類以下に限定すれば、1対1に変換可能であることが知られている。これについては、先の非特許文献4が詳しい。従って、実施の形態3における配列出力部300は、整数Xに対して1対1に対応する配列V40を出力している。そのため、配列出力部300は、入力された整数Xに基づいて一様に配列を出力していることになる。

【0229】ここで、実施の形態3と、実施の形態1、2の効果との差異を説明する。整数Xの値が、実施の形態1では、一つの出力値に対して、入力値が( $n1$ )! $\times$ ( $n2$ )! $\times$ ( $n-n1-n2$ )!、実施の形態2では、( $n1$ )! $\times$ ( $n2$ )!種類存在する。それに対して、実施の形態3では、1対1で変換する。そのため、入力の様性を保持するために必要な入力のビット長が最小になっている。

【0230】実際のパラメータの場合について説明する。 $n=263$ 、 $n1=16$ 、 $n2=16$ とする時、 $\phi(n, n1, n2)$ に属する配列は、 $n! / ((n1)! \times (n2)! \times (n-n1-n2)!)$   $\div 2^{163}$ であるので、ハッシュ関数の出力長は、163ビット以上でよく、これは、実施の形態2に比べて、92ビット短い。

【0231】しかし、実施の形態3の配列出力装置300の第1の数配置部、第2の数配置部では、 $C(n-n1, c2)$ や $C(n-n1-n2, c2)$ の計算を行う必要がある。この計算は、階乗の計算を含むため、計算量が大きくなる。一方、実施の形態1、2では、階乗の計算を行わないため、計算量が小さい。

【0232】このように、本実施の形態によれば、配列出力部300は、整数Xに基づいて一様にn次元の配列を出力することができる。そのため、NTRU暗号にFOSRTを適用して、図1(a)の暗号化装置10の配列出力部100の代わりにこの配列出力部300とし、ハッシュ関数部40より出力されるハッシュ関数値H(m)に基づいて、配列出力部300が一様にn次元の配列を出力することで、ハッシュ関数の分布の様性を保持することが可能となり、暗号化装置10で生成する暗号文の安全性を高めることができる。

【0233】さらに、配列出力部300は、整数Xのみの情報より配列V40を設定しているため、メモリテーブルを用いる必要がなく、メモリが少なくてよいという効果が得られる。

【0234】また、図1(b)の復号化装置15の配列出力部105の代わりに、この配列出力部300を用いることで暗号化された暗号文の復号が可能となる。

【0235】尚、また、配列出力部300を暗号化装置10に用いて、ハッシュ関数値を基に配列を出力する構成としたがこれに限定されない。

【0236】(実施の形態4)本発明における第4の実施の形態に係る暗号化装置を説明する。本実施の形態の暗号化装置は、図1の暗号化装置10と比べて配列出力部100が異なる構成の配列出力部400となっている。その他の構成は共通しているため、その説明を省略する。

【0237】本実施の形態における配列出力部400を、図面を用いて説明する。図14は、本実施の形態における配列出力部400の構成を示すブロック図である。

【0238】この配列出力部400は、整数Xを入力とし、 $L(n, n1, n2)$ に属する配列V50を出力するものである。ここで、 $L(n, n1, n2)$ は、 $n1$ 個の要素が1、 $n2$ 個の要素が-1、その他の( $n-n1-n2$ )個の要素が0であるn次元配列全体であり、 $n, n1, n2$ は配列出力部400に予め設定されているものとする。この配列出力部400は、すべての配列要素が0の配列要素を暫定的に決定し、0の配列要素を整数Xに基づいて1、および-1に改竄する。

【0239】配列出力部400は、配列出力部100と同様、マイクロコンピュータのソフトウェア、あるいは電子回路などのハードウェアにより処理を実行する。

【0240】次に、配列出力部400の動作を説明する。まず、配列出力部400は、配列V50をすべての要素が0である配列状態にする。

【0241】次に、配列出力部400は、整数Xを8ビット毎に分割する。整数Xは0、1の2値で表されたビット情報の集まりで示されており、図16に示すように、整数Xは8ビット毎に( $n1+n2$ )個に分割される。

【0242】図16は、整数Xを各分割情報[0]、分割情報D[1]～分割情報D[ $n1+n2-1$ ]に分解した状態を示す図である。各分割情報D[0]、分割情報D[1]～分割情報D[ $n1+n2-1$ ]は、それぞれ8ビットの情報により整数を示す。

【0243】ここで、8ビットの分割情報D[0]が整数Qを示しているとする、次に配列出力部400は、配列V50において、Q+1番目(以下、p0番目という)の要素が0である場合に1に設定する。つづいて、配列出力部400は、 $p1 = (p0 + D[1]) \bmod n$ で示されるp1番目の要素が0である場合に1に設定する。

【0244】このように、配列出力部400は、分割情報D[i]を基に、配列V50の $p_i = (P(i-1) + D[i]) \bmod n$ (ここで、 $i=1 \sim n1+n2-1$ )で示されるp\_i番目の要素が0である場合に順に1に設定する。また、配列V50のp\_i番目の要素と



は配列V50の左から $p_i$ 番目の要素をいう。

【0245】このとき、配列出力部400は、配列V50の $p_i$ 番目の要素が0でない場合は1に設定せず、 $p_i \leftarrow (p_i + 1) \bmod (n)$ とし、 $p_i$ 番目の右側にある0の要素を1に設定する。

【0246】配列出力部400は、このように配列V50の要素を1に設定する処理を行い、1の要素が $n-1$ 個になると、次に同様に分割情報D[i]を基に、配列V50の-1の要素が $n-2$ 個になるまで-1の要素を設定する処理を行う。

【0247】ここで配列出力部400の詳細な動作を説明する。図15は、配列出力部400が行う処理を示すフローチャートである。配列出力部400は、以下のようなステップの処理で行われる。なお、カウンタ $c_1'$ の値をカウント値 $c_1$ 、カウンタ $c_2'$ の値をカウント値 $c_2$ とする。

【0248】まず配列出力部400は、変数Y10に整数Xを代入する(ステップS601)。次に、配列出力部400は、配列V50を全ての要素が0である配列状態とする(ステップS602)。

【0249】次に、配列出力部400は、整数Xを8ビット毎に区切って分割情報D[0]、分割情報D[1]〜分割情報D[n1+n2-1]に分割する(ステップS603)。

【0250】次に、配列出力部400は、カウンタ $c_1'$ のカウント値 $c_1$ を0にする(ステップS604)。

【0251】次に、配列出力部400は、カウンタ $c_2'$ のカウント値 $c_2$ をD[0]+1とする(ステップS605)。すなわち、カウント値 $c_2$ は、8ビットの分割情報D[0]で示される整数Q+1の値となる。

【0252】次に、配列出力部400は、配列V50の $c_2$ 番目の要素V50[c2]が0であるかを判定し(ステップS606)、0でないとき判定した時は(ステップS606のNo)、カウント値 $c_2$ を $c_2 \leftarrow (c_2 + 1) \bmod (n)$ とし(ステップS607)、再度、要素V50[c2]が0であるか否かの判定を行う(ステップS606)。一方、配列出力部400は、V50[c2]が0であると判定した時は(ステップS606のYes)、要素V50[c2]を1に設定する(ステップS608)。

【0253】これらの要素V50[c2]が0であるかの判定処理(ステップS606)、 $c_2 \leftarrow (c_2 + 1) \bmod (n)$ の処理(ステップS607)、およびV50[c2]-1の処理(ステップS608)により、V50[c2]の要素が0でない場合に、V50[c2]の右側の要素で0であるものを巡回移動しながら順に探して1に設定する。

【0254】そして次に、配列出力部400は、カウンタ $c_1'$ のカウント値 $c_1$ が $c_1 < n-1$ であるかの

判定を行う(ステップS609)。

【0255】配列出力部400は、 $c_1 < n-1$ であると判定したとき(ステップS609のYes)、カウント値 $c_1$ をカウントアップすると共に( $c_1 \leftarrow c_1 + 1$ )、カウント値 $c_2$ を $c_2 \leftarrow (c_2 + D[c_1]) \bmod (n)$ とし(ステップS610)、そして再度、V50[c2]が0であるかの判定を行う(ステップS606)。

【0256】このように、配列V50の1の要素が $n-1$ 個になるまで、1の要素の設定処理が行われる。

【0257】一方、配列出力部400は、 $c_1 < n-1$ でないと判定したとき(ステップS609のNo)、配列V50の1の要素が $n-1$ 個になったとして、配列V50における-1の要素の設定処理に移るべく、カウント値 $c_1$ を0とし(ステップS611)、カウント値 $c_2$ を $c_2 \leftarrow (c_2 + D[n-1]) \bmod (n)$ とする(ステップS612)。

【0258】次に、配列出力部400は、配列V50の要素V50[c2]が0であるかを判定し(ステップS614)、0でないとき判定した時は(ステップS614のNo)、カウント値 $c_2$ を $c_2 \leftarrow (c_2 + 1) \bmod (n)$ とし(ステップS613)、再度、要素V50[c2]が0であるか否かの判定を行う(ステップS614)。一方、配列出力部400は、V50[c2]が0であると判定した時(ステップS614のYes)、要素V50[c2]を-1に設定する(ステップS615)。

【0259】これらの要素V50[c2]が0であるかの判定処理(ステップS614)、 $c_2 \leftarrow (c_2 + 1) \bmod (n)$ の処理(ステップS613)、およびV50[c2]-1の処理(ステップS615)により、V50[c2]の要素が0でない場合に、V50[c2]の右側の要素で0であるものを巡回移動しながら順に探して-1に設定する。

【0260】そして次に、配列出力部400は、カウンタ $c_1'$ のカウント値 $c_1$ が $c_1 < n-1$ であるかの判定を行う(ステップS616)。

【0261】配列出力部400は、 $c_1 < n-1$ であると判定したとき(ステップS616のYes)、カウント値 $c_1$ をカウントアップすると共に( $c_1 \leftarrow c_1 + 1$ )、カウント値 $c_2$ を $c_2 \leftarrow (c_2 + D[c_1 + n-1]) \bmod (n)$ とし(ステップS617)、そして再度、V50[c2]が0であるかの判定を行う(ステップS614)。

【0262】一方、配列出力部400は、 $c_1 < n-1$ でないと判定したとき(ステップS616のNo)、配列V50の-1の要素が $n-2$ 個になったとして配列V50を出力する。

【0263】次に、図15のフローチャートにそって具体例を説明する。図17は配列出力部400における配



列V50の各段階での配列状態を示している。

【0264】配列V50は251次元の配列( $n=251$ )であり、 $n1=50$ 、 $n2=50$ とする。また、整数 $X$ を8ビットごとに分割したときの分割情報を例えば、分割情報 $D[0]=139$ 、分割情報 $D[1]=130$ とする。

【0265】図15のフローチャートによる流れで、本具体例に従えば、カウント値 $c2=D[0]+1=140$ となる(ステップS605)。

【0266】そして、配列V50は全要素が0であるため、配列V50の左から140番目の要素V50[140]を1に設定する(ステップS608)。この配列状態は、図17に示す配列V51の通りであり、左から140番目の要素が1であり、その他の要素が0となっている。

【0267】次に、配列V50の1の要素は50個でないため(ステップS609のNo)、配列V50の要素を1に設定する処理をさらに行うため、カウント値 $c2$ を $c2 \leftarrow (c2 + D[1]) \bmod (n)$ とする(ステップS610)。カウント値 $c2=(140+130) \bmod 251=19 \bmod 251$ となり、配列V51の左から19番目の要素V51[19]を1にする。この配列状態は、図17に示す配列V52の通りであり、左から19番目と140番目の要素が1であり、その他の要素が0となっている。

【0268】このように、配列V50の各要素に1を設定する処理をするとき、設定する位置の要素、例えばV50[120]がすでに1となっている場合は、その右側の要素を巡回移動して0の要素を探し、一番初めの0の要素を1に設定する。そして、V50[251]までのすべての要素が0でない場合は、巡回移動して一番左のV50[1]の要素に戻り、0の要素を1に設定する。同様にして、配列V50の-1の要素を順に設定する。

【0269】上述したように、配列出力部400は、配列V50における最初の1の要素を整数 $X$ より求まる分割情報 $D[0]$ に基づいて一様に決定する。そして、最初に決定された1の要素の場所から整数 $X$ により求まる分割情報 $D[i]$ を基に順に次の要素を設定する位置を決定して、1あるいは-1を設定するため、配列V50は、整数 $X$ から一様に分布する。

【0270】このとき、配列出力部400は、配列V50における $n1$ 個の1の要素と、 $n2$ 個の-1の要素を決定するために、入力される整数 $X$ は、8ビットの分割情報を( $n1+n2$ )個必要である。そのために、整数 $X$ は、配列V50の各要素を十分設定できる大きいものを設計段階で選択すればよい。

【0271】このように、本実施の形態によれば、配列出力部400は、整数 $X$ に基づいて、一様に $n$ 次元の配列を出力することができる。そのため、NTRU暗号に

FOSRTを適用した場合において、図1(a)の暗号化装置10の配列出力部100の代わりにこの配列出力部400とし、ハッシュ関数部40より出力されるハッシュ関数値 $H(m)$ に基づいて、配列出力部400が一様に $n$ 次元の配列を出力することで、ハッシュ関数の分布の一様性を保持することが可能となり、暗号化装置10で生成する暗号文の安全性を高めることができる。

【0272】さらに、配列出力部400は、整数 $X$ のみの情報より配列V50を設定しているため、メモリテーブルを用いる必要がなく、メモリが少なくてもよいという効果が得られる。

【0273】また、図1(b)の復号化装置15の配列出力部105の代わりに、この配列出力部200を用いることで暗号化された暗号文の復号が可能となる。

【0274】尚、配列出力部100を暗号化装置10に用いて、ハッシュ関数値を基に配列を出力する構成としたがこれに限定されない。

【0275】また、各実施の形態で説明した暗号化装置10は、図18に示す携帯電話機500内に搭載されて使用されたり、また、インターネット上の電子決済、電子商取引として使用されたりする。

【0276】また、各実施の形態1、2、3、4において、各配列出力部は、 $n1$ 個の要素が1、 $n2$ 個の要素が-1、その他の要素が0の配列を出力しているが、1、-1が他の数であってもよい。また、各実施の形態1、2、3、4において、各配列出力部は、1、-1、0の3値の配列を出力するが、2値であってもよいし、4値以上であってもよい。

【0277】また各実施の形態1、2、3、4のうちのいずれかを使用する暗号方法であってもよい。

【0278】

【発明の効果】以上の説明から明らかなように、本発明に係る配列出力装置においては、入力された整数に依って、 $K$ 値の整数の組み合わせからなる $n$ 次元の様々な配列を出力する配列出力装置であって、初期配列を暫定的に決定する初期配列決定手段と、前記入力された整数に基づいて、前記初期配列決定手段が決定した前記初期配列の配列要素を改編する改編手段とを備えることを特徴とする。

【0279】これによって、多くのメモリを使用することなく、整数値に基づいて一様に $n$ 次元の配列を得ることができ、ハッシュ関数値などの一様に分布された整数値に基づいてその一様性を保持して $n$ 次元の配列を得ることが可能となる。

【0280】また、前記改編手段は、前記入力された整数を所定の整数で除算し剰余を求める除算部と、前記除算部が求めた剰余に基づいて、前記初期配列の配列要素を置換する置換部とを備えるようにしてもよい。

【0281】これにより、ハッシュ関数値などの一様に分布された整数値に基づいてその一様性を保持して $n$ 次



元の配列を得ることが可能となる。

【0282】また、前記改編手段は、前記入力された整数を所定の整数で除算し剰余を求める除算部と、前記初期配列における整数P3の配列要素のうち前記除算部が求めた剰余に基づく位置の配列要素を整数P1に置き換える整数配置部とを備えるようにしてもよい。

【0283】これにより、ハッシュ関数値などの一様に分布された整数値に基づいてその一様性を保持してn次元の配列を得ることが可能となる。

【0284】また、メッセージを暗号化する暗号化装置であって、メッセージを一方向関数で演算し、その結果を関数値として出力する関数値出力手段と、初期配列を暫定的に決定する初期配列決定手段、および前記関数値出力手段が出力する関数値に基づいて、前記初期配列決定手段が決定した前記初期配列の要素を改編する改編手段を備え、前記関数値に依存して、K値の整数の組み合わせからなるn次元の様々な配列を出力する配列出力手段と、前記配列出力手段が出力する配列を基に、暗号文を生成する暗号文生成手段とを備えることを特徴とする。

【0285】これにより、メッセージ値のハッシュ関数値などの、一方向関数により一様に分布された整数値に基づいて、その一様性を保持してn次元の配列を得ることが可能となり、暗号文の安全性を高くすることができる。

【図面の簡単な説明】

【図1】本発明における第1の実施の形態に係る(a)は暗号化装置の構成を示すブロック図であり、(b)は復号化装置の構成を示すブロック図である。

【図2】同上の配列出力部の構成を示すブロック図である。

【図3】同上の配列出力部から出力される配列の配列状態を示す図である。

【図4】同上の配列出力部の動作を示すフローチャートである。

【図5】本発明における第2の実施の形態に係る配列出力部の構成を示すブロック図である。

【図6】同上の配列出力部における第1の数配置部の動作を示すフローチャートである。

【図7】同上の配列出力部における第1の数配置部から出力される配列の配列状態を示す図である。

【図8】同上の配列出力部における第2の数配置部の動作を示すフローチャートである。

【図9】同上の配列出力部における第2の数配置部から出力される配列の配列状態を示す図である。

【図10】本発明における第3の実施の形態に係る配列出力部の構成を示すブロック図である。

【図11】同上の配列出力部における第1の数配置部から出力される配列の配列状態を示す図である。

【図12】同上の配列出力部における第2の数配置部から出力される配列の配列状態を示す図である。

【図13】(a)(b)(c)はいずれも同上の配列出力部の各部で設定される配列の配列状態を示す図である。

【図14】本発明における第4の実施の形態に係る配列出力部の構成を示す図である。

【図15】同上の配列出力部の動作を示すフローチャートである。

【図16】同上の配列出力部に入力される整数を示す図である。

【図17】同上の配列出力部から出力される配列の配列状態を示す図である。

【図18】本発明の配列出力部を有する暗号化装置が搭載される携帯電話機の外観を示す外観図である。

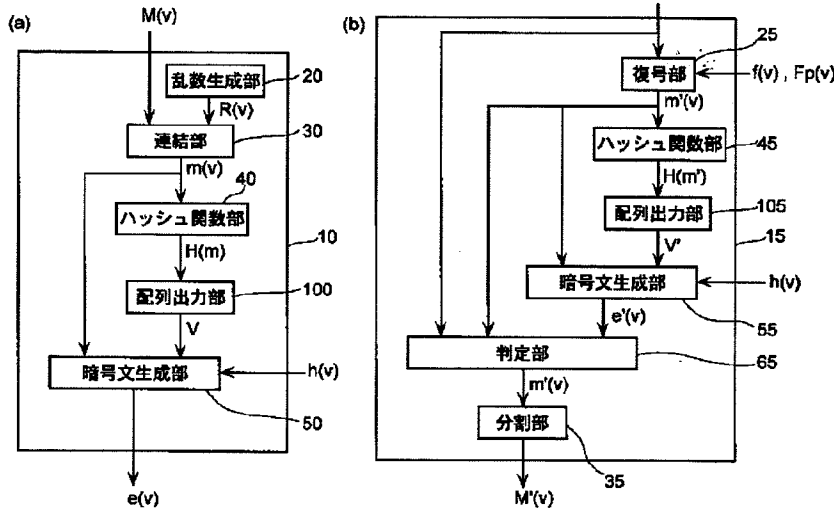
【図19】従来の配列出力方法を示すフローチャートである。

【符号の説明】

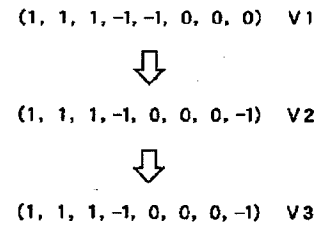
10	暗号化装置
15	復号化装置
20	乱数生成部
25	復号部
30	連結部
35	分割部
40	ハッシュ関数部
45	ハッシュ関数部
50	暗号文生成部
55	暗号文生成部
65	判定部
100	配列出力部
110	初期配列決定部
120	配列要素置換部
200	配列出力部
210	第1の数配置部
220	第2の数配置部
300	配列出力部
310	第1の数配置部
320	第2の数配置部
400	配列出力部
500	携帯電話機



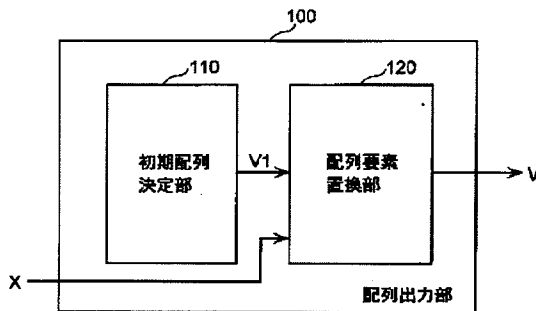
【図1】



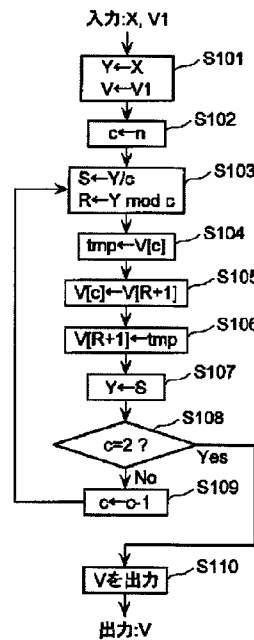
【図3】



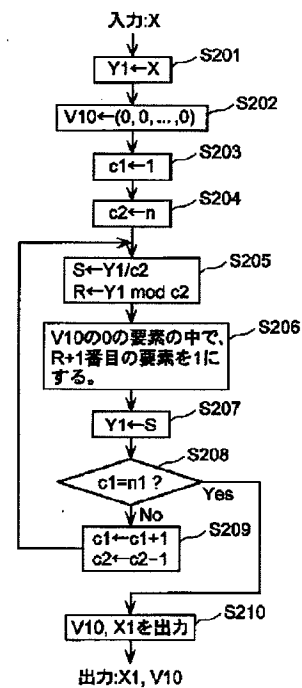
【図2】



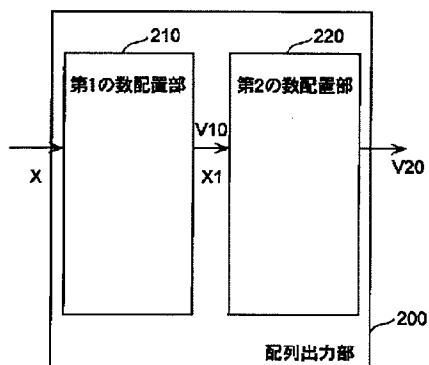
【図4】



【図6】

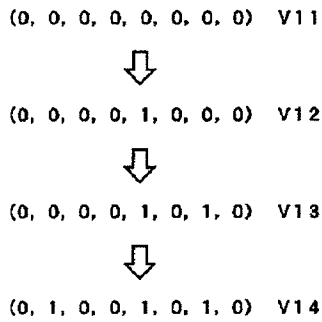


【図5】

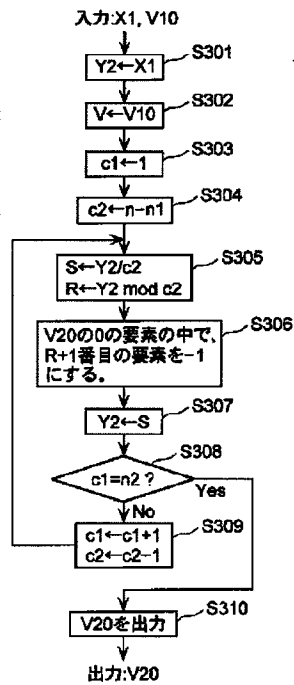




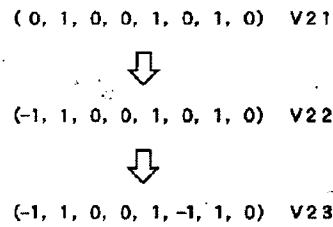
【図7】



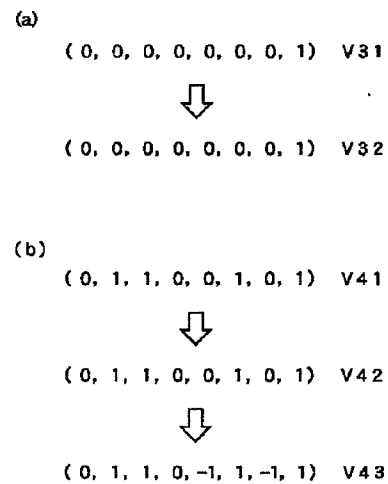
【図8】



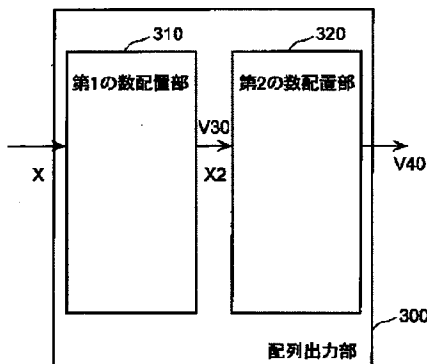
【図9】



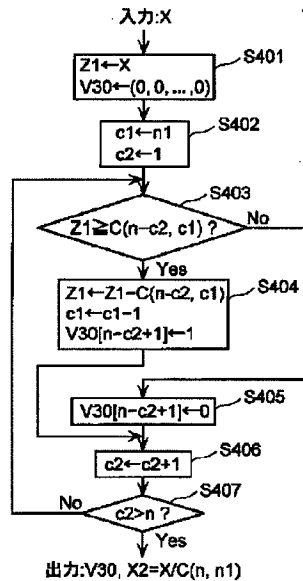
【図13】



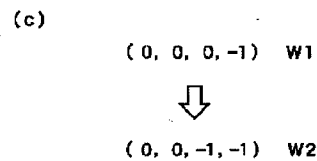
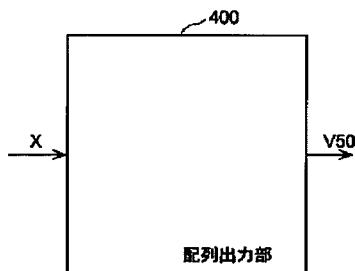
【図10】



【図11】

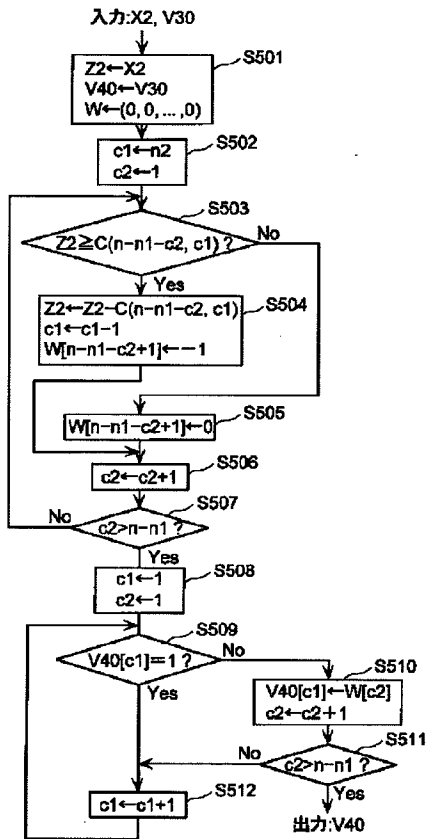


【図14】

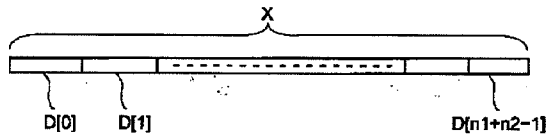




【図12】



【図16】

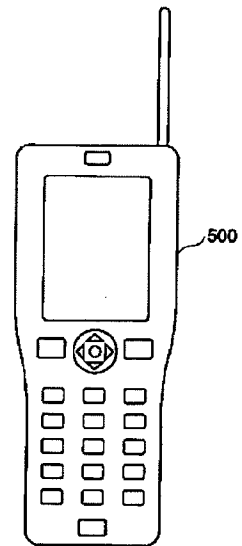


【図17】

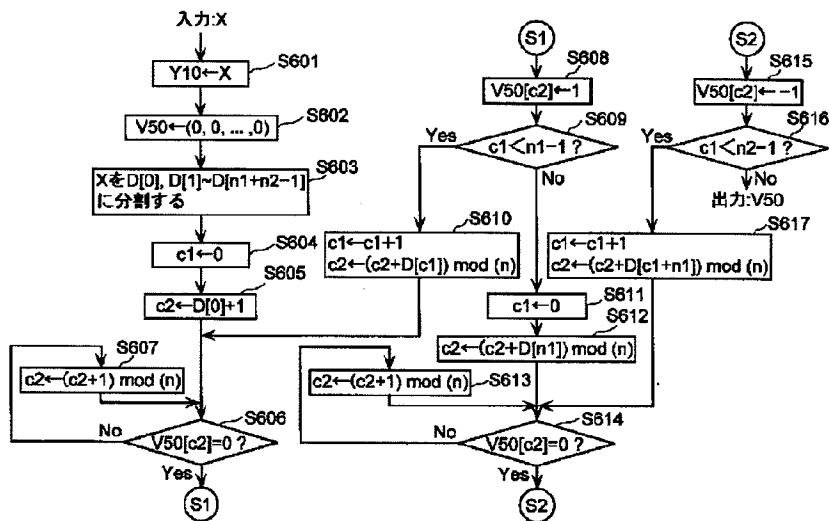
(0, 0, 0, 0, ..., 0, 1, 0, ..., 0) V51

(0, 0, 1, 0, ..., 0, 1, 0, ..., 0) V52

【図18】

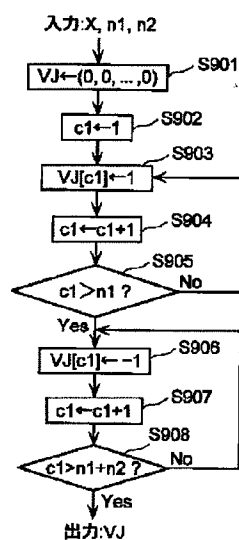


【図15】





【図19】




---

フロントページの続き

(72)発明者 横田 薫

大阪府門真市大字門真1006番地 松下電器  
産業株式会社内

(72)発明者 館林 誠

大阪府門真市大字門真1006番地 松下電器  
産業株式会社内

Fターム(参考) 5J104 AA18 JA21 NA12 NA39